ROBUST TARGET LOCALIZATION AND SEGMENTATION USING STATISTICAL METHODS

A Thesis
Presented to
The Academic Faculty

by

Omar Arif

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
May 2010

ROBUST TARGET LOCALIZATION AND SEGMENTATION USING STATISTICAL METHODS

Approved by:

Dr. Allen R Tannenbaum,
Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Aaron D Lanterman
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Patricio Antonio Vela, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Jochen Teizer
School of Civil and Environment Engineering
Georgia Institute of Technology

Dr. Anthony Joseph Yezzi
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: April 2, 2010

To my parents.

ACKNOWLEDGEMENTS

My deep and sincere gratitude to my advisor Dr. Patricio Vela to whom I will always be grateful. I thank him for all the guidance and support that he has provided all these years. I also thank Dr. Allen Tannenbaum, Dr. Anthony Yezzi, Dr. Aaron Lanterman and Dr. Jochen Teizer for serving on my PhD dissertation defense committee.

Many thanks to my lab-mates: Ibrahima Ndiour, Miguel Serrano, GBolabo Ogunmakin and Jun Yang. I also wish to thank former and current Pakistani students at Georgia Tech. Special thanks to Manzar Abbas, Messam Naqvi and Murtaza Askari for helping me settle down initially. Thanks to Tauseef-ur-Rehman, Taimoor Khawaja, Ali Hashmi, Aleem, Farooq Akram, Salman, Qaisar Chaudary, and Irtezaa for their help and support. Many thanks to my old friend Uzair Hashmi, who remained in contact with me all these years.

My heartfelt gratitude to my father and mother. I thank them for all the love, guidance and prayers. I am blessed to have such a loving parents. My love and thanks also goes to my brothers Bilal, Saad and Jamal.

My deepest love and affection to my wife for all the support, love and care. Her company made this duration a pleasant and a joyous one. Also, love to my kids Aizah and Hafsah, who were a source of happiness and blessings.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis aims to contribute to the area of visual tracking, which is the process of identifying an object of interest through a sequence of successive images. Some of the challenges associated with these tasks are image noise, occlusions, background clutter, complex object shapes, etc.

The work contained in this thesis explores kernel-based statistical methods. These methods map the data to a higher dimensional space where the tasks of classification and clustering are easily carried out. There are two problems related to the mapping: The out-of-sample and the pre-image problem. A pre-image framework for some of the manifold learning and dimensional reduction methods is developed.

Two algorithms are developed for visual tracking that are robust to noise and occlusions. In the first algorithm (Chapter 3), a KPCA-based eigenspace representation is used. The de-noising and clustering capabilities of the KPCA procedure lead to a robust algorithm. This framework is further extended in Chapter 6 to incorporate the background information in an energy based formulation, which is minimized using graph cut. Chapter 7 extends this framework to track multiple objects using a single learned model.

In the second method, a robust density comparison framework is developed (Chapter 5) that is applied to visual tracking (Chapter 8), where an object is tracked by minimizing the distance between a model distribution and given candidate distributions.

The superior performance of kernel-based algorithms comes at a price of increased storage and computational requirements. A novel method is proposed in Chapter 4, that takes advantage of the universal approximation capabilities of generalized radial basis function neural networks to reduce the computational and storage requirements for kernel-based methods. The ideas developed are general and are applicable to other kernel-based methods, such as KCPA and support vector machines.

CHAPTER I

INTRODUCTION AND BACKGROUND

Computer vision aims to artificially replicate the human visual perceptions. It deals with the science and technology of processes related to the acquisition of images, analysis of images and sequence of images to extract useful information, and to the development of artificial cognitive systems that "see."

Image segmentation and visual tracking are two important components of computer vision. The former deals with an image, while the latter is concerned with the sequence of images. Segmentation aims to partition an image into smaller more meaningful parts, which are related with respect to some common aspect. Visual tracking, on the other hand, is the process of locating an object of interest in a sequence of successive images. For a deformable object, the motion of the object, over a sequence of images, is described by an overall global motion (pose), and the local deformation of the object [94]. In this respect, at each frame of the sequence, segmentation is required to delineate the target from the background. Tracking a deformable object is therefore, the process of estimating the pose parameters (*target localization*) and the local deformation (*target segmentation*) of the object. In this sense, visual tracking encompasses segmentation. Algorithms differ in whether they perform localization only or both i.e. localization and segmentation. The former can be called *transformation based* and the latter *contour based*. Objects can be represented by their appearances such as color, texture, edges etc, which provide characteristic information about the target object. This characteristic information is encoded into a cost or similarity functional. Tracking algorithms then find the target object that is optimum with respect to a pre-determined similarity functional.

Visual tracking is a challenging task. In some cases, the image information (appearance) may not be sufficient enough to identify the target object. This may happen due to a number of reasons, such as noise, occlusions, complex object shapes and presence of target

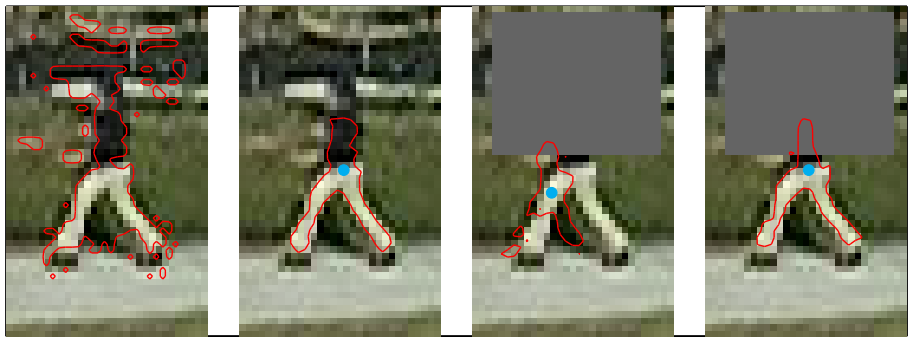
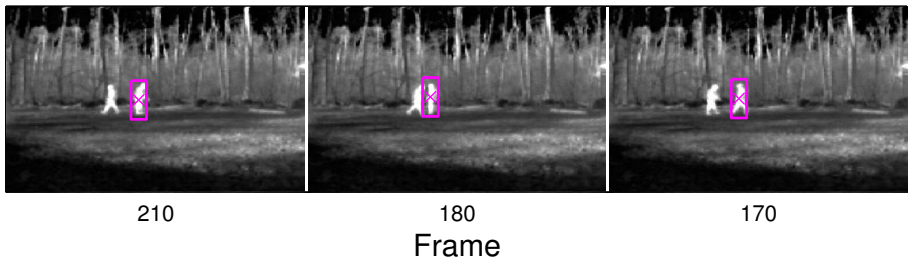


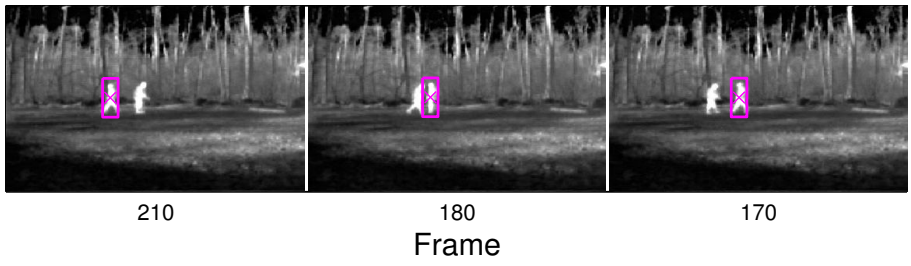
Figure 1: Segmentation: without shape prior, with shape prior, shape prior and occlusion, shape prior and occlusion with manual localization (from left to right).

features in the background etc. In such cases, visual tracking can be simplified by imposing a shape/spatial constraint on the tracking process. These constraints can come in the form of smoothness priors on the contour that segments the object from the background, or in the form of incorporating prior known information on the shape of the object to be segmented. For example, in the first image in Figure 1, the target object is segmented without any shape prior. The segmentation is noisy, due to the presence of the target features (color) in the background. The second image produces a correct segmentation by incorporating shape priors. However, the addition of shape information requires extra effort on the part of tracker. The prior shapes need to be aligned with the pose/location of the object, which is being tracked, for correct segmentation. This is not trivial since the pose is not known and has to be estimated along with the segmentation. If the pose can not be adequately estimated, then the tracking results are meaningless. The effect of incorrect target localization on target segmentation is shown in Figure 1-third image. In the last image, the target is localized manually, which results in correct segmentation. So correct target localization is critical to shape based trackers.

Apart from imposing constraints on the shape of the object, tracking can also be simplified by imposing constraints on object motion. The object motion can be constrained to be constant velocity or constant acceleration based on prior information [110]. Figure 2 shows an example of successful tracking using a constant velocity motion constraint. These methodologies form the basis of well known tracking algorithms like Kalman filter



(a) Without motion constraint



(b) With constant velocity constraint

Figure 2: Tracking with and without Kalman filtering

[50] and particle filter [44]. These techniques have been used in tracking algorithms such as [18, 28, 49, 101]. Filtering techniques will not be pursued in this research.

The computation complexity of visual tracking algorithms is an important issue. Tracking algorithms should be able to work in real time.

This thesis aims to contribute to the above mentioned challenges and requirements for visual tracking. It does this by developing novel algorithms for target localization and segmentation, which are robust to noise and massive occlusions. The algorithms are based on kernel-based statistical methods. Also, a novel method is provided to reduce the computation complexity of the algorithms.

Below, target segmentation and localization, as carried out by different methods, is explained briefly. We will be mostly concerned with how shape information is incorporated and used to perform localization and segmentation.

1.1 Target Segmentation

The aim of target segmentation is to delineate the object from its background. Several algorithms and techniques have been developed for segmentation. The more recent ones are discussed below.

1.1.1 Active Contour based

The basic idea in active contour based segmentation algorithms is to evolve a closed contour, starting from an initial estimate, such that it encircles the object region. Evolution of the contour is governed by an energy functional, which defines the fitness of the contour to the hypothesized object region [110]. The energy functional is minimized when the contour delineates the object from the background. The energy functional can be based on local information such as the image gradient [52, 54], global features such as color [41, 109, 114], or mean image intensity inside and outside of the evolving contour [25]. Level set methods [89] have been successfully used for implicit representation of the contour. The most important advantage of level set representation of the contour is its flexibility in allowing topology changes. There is a vast body of literature concerning level sets and active contour, see for example [68, 69, 70, 78].

More recently, shape information is incorporated into the active contour framework to make the segmentation robust to noise and occlusion [33, 34, 35, 40, 60, 79, 98]. Leventon et al. [60] define the shape term E_{shape} using a probabilistic approach based on principal component analysis (PCA). The set of training shapes are represented by their signed distance functions [89], and PCA is applied to obtain a reduced representation. A probability density function is defined over the parameters of the reduced representation to obtain the shape energy. The level set function is evolved using both the image and the shape terms, which draw the level set function towards the most probable shape according to the learned distribution. Tsai et al. [98] incorporate the shape model, derived also by performing PCA on a collection of signed distance maps of the training shapes, into region-based active contour ([25]). The problem is reformulated to directly optimize the parameters associated with the first few eigenvectors. Cremers et al. [33] use kernel density estimator to define the probability density on the space of signed distance function representing the prior shapes. They show that this approach captures nonlinear shape variability. Freedman et al. [40] track by combining density matching and shape priors. For density matching, Bhattacharyya measure is used to define the distance between a model intensity distribution and the intensity distribution of an estimated image region. The tracker is expressed as a PDE-based curve

evolution, which is implemented using level sets. Dambreville et al. [35] combine intensity based segmentation with prior shape knowledge learned using Kernel PCA (KPCA). A binary representation of shapes is used. KPCA is shown to outperform linear PCA, by allowing only shapes that are close enough to the training data. Dynamical shape priors have also been used to improve the tracking of deformable objects in the presence of noise and occlusions. [30, 77].

1.1.2 Graph cut

Image segmentation can also be formulated as a graph partitioning problem [23]. Let $\mathcal{R}$ be the set of all pixels in the image, and let $\mathcal{L} = \{0, 1\}$ be a label assignments on $\mathcal{R}$. The label 1 means the pixel belongs to the target, while the label 0 means it belongs to the background. The segmentation problem is cast as that of finding a labeling $l : \mathcal{R} \rightarrow \mathcal{L}$, minimizing an energy $E(l)$, modeled by:

$$E(l) = E_d(\mathcal{I}, l) + E_s(l). \tag{1}$$

E_d is the data term which measures how well the labeled pixels fit the image model. The standard data term is

$$E_d(\mathcal{I}, l) = \sum_{u \in \mathcal{R}} F_u(l_u), \tag{2}$$

where F_u measures how well label l_u fits pixel u. To realize the energy on the graph, each pixel is considered a node of the graph with two additional terminal nodes, the target and the background terminal nodes. To encode the data term on the graph, each pixel is connected to the target and background terminal nodes with edge weights $F_u(1)$ and $F_u(0)$, representing the cost of assigning a pixel to target and background respectively. E_s is the regularization or boundary term. Let $\mathcal{N}$ be a neighborhood system on $\mathcal{R}$, then E_s is realized by connecting each pair of pixels $(u, v) \in \mathcal{N}$ with a non-negative edge weight measuring the penalty for assigning two neighboring pixels to different regions. The mincut of the graph represents the segmentation that best separates the target from its background and minimizes the energy $E(l)$.

Prior shape information can be encoded in the graph cut by either imposing it on the edges between pixels and the terminals nodes [63, 105], or by defining the edge weights

between neighboring pixels [42, 55, 100]. Vu and Manjunath [105] use shape distance as shape penalty and impose it on the terminal edges. Shape alignment is done intrinsically from the shape's moment, as is done in [63, 42]. Veksler [100] uses a star shape prior with manual registration, i.e., it is assumed that the center of the star shape is known or provided by the user. Malcolm et al. [65] use distance penalty from the location of the target object to impose additional penalty on the terminal edges. A filter is also employed which predicts the location of the target object and the distance penalty is then centered at that location in the next frame.

1.2 Target Localization

As mentioned before, the estimation of the pose of the target object is critical to all trackers. Different methods employ different procedures to estimate the pose of the object from frame to frame. Some algorithms estimate the pose parameters and perform segmentation simultaneously, while others formulate it as a two step approach. First they estimate the pose and then perform segmentation.

Leventon et al. [60] use maximum a posteriori approach to estimate the pose of the evolving contour, based on the prior shape information. Tsai et al. [98] estimate the pose parameters using gradient descent on the objective function for segmentation. The target localization can be made invariant to pose parameters by an intrinsic registration, such as moments, of the evolving contour [30, 33, 63, 105]. The prior shapes are assumed to be registered intrinsically. The evolving contour is also aligned intrinsically by computing a transformation T. The prior shapes are then aligned with the contour by reversing the transformation. This procedure removes the need to iteratively optimize explicit pose parameters.

As mentioned before, there are transformation based tracking algorithms that find the global transformation of the object of interest in consecutive frames. Segmentation is not carried out. The output of such trackers is the estimated pose parameters of the target object. We leave the discussion of the transformation based trackers to Chapter 3.

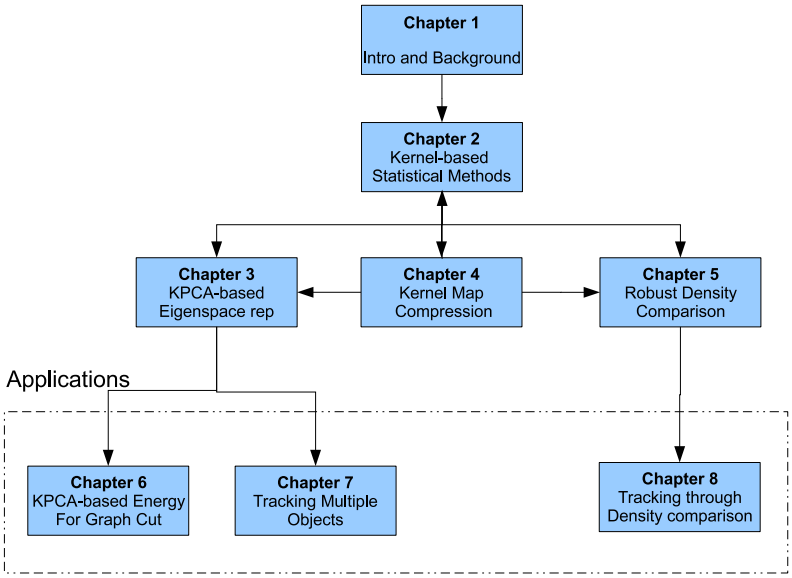


Figure 3: Outline of the thesis

1.3 Oganization and Contributions of the thesis

The outline of the thesis can be understood from Figure 3, where each chapter represents a block. The arrows indicate the hierarchy of the contributions in each chapter. The organization and the contributions are as follows:

Chapter 2: Describes kernel-based statistical methods used in component analysis, dimensional reduction and manifold learning . These methods map the data to a higher dimensional space where the tasks of classification and clustering are easily carried out. There are two problems related to the mapping: The out-of-sample and the pre-image problem. The contribution of the chapter is in providing the pre-image methods for some of the manifold learning and dimensional reduction methods.

Chapter 3: Presents a non-rigid object localization and segmentation algorithm using a KPCA-based eigenspace representation. Localization and segmentation are carried out by deriving a similarity function in the KPCA eigenspace. The KPCA space is related non-linearly to the input space, which results in a non-linear algorithm. The de-noising and clustering capabilities of the KPCA procedure lead to a localization procedure that is robust

to noise and occlusions. A unique feature of the approach is that it permits segmentation in addition to localization when multiple templates of the target are given.

Chapter 4: The computational complexity of the kernel-based statistical methods is of the order of the training set, which is quite large for many applications. This chapter proposes a two step procedure for arriving at a compact and computationally efficient learning procedure. After learning, the second step takes advantage of the universal approximation capabilities of generalized radial basis function neural networks to efficiently approximate the empirical kernel maps. The ideas developed in this chapter are used to reduce the computational complexity of the proposed methods in Chapter 2, 3 and 5.

Chapter 5: This chapter presents a technique to robustly compare two distributions represented by samples, without explicitly estimating the density. The method is based on mapping the distributions into a reproducing kernel Hilbert space, where eigenvalue decomposition is performed. Retention of only the leading m eigenvectors minimizes the effect of noise on density comparison.

Chapter 6: This chapter extends the tracking method presented in Chapter 3 by incorporating the background information. Energy based formulation is used, which is minimized using graph cut.

Chapter 7: This chapter extends the tracking framework developed in Chapter 3 to track multiple objects.

Chapter 8: This chapter is an application of the robust density comparison technique developed in Chapter 5. The robust density comparison is applied to target localization, where the object to be tracked is assumed to be characterized by a probability density. To track the object, a gradient based search criteria is developed to find the region whose sample distribution closely matches the model distribution.

This thesis is a combination of the following papers [5, 6, 7, 8, 9, 10, 11, 102].

CHAPTER II

KERNEL-BASED STATISTICAL METHODS

2.1 Mercer kernels

Let $\{u_i\}_{i=1}^n, u_i \in \mathbb{R}^d$, be a set of n observations, a Mercer kernel is a function $\boldsymbol{k} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$, which satisfies:

1. $\boldsymbol{k}$ is continuous
2. $\boldsymbol{k}(u_i, u_j) = \boldsymbol{k}(u_j, u_i)$. Symmetric
3. The matrix K, with entries $K_{ij} = \boldsymbol{k}(u_i, u_j)$ is positive definite.

Theorem: If $\boldsymbol{k}$ is a Mercer kernel then there exists a high dimensional Hilbert space $\mathcal{H}$, with mapping $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$ such that:

$$\phi(u_i) \cdot \phi(u_j) = \boldsymbol{k}(u_i, u_j). \tag{3}$$

The mercer kernel $\boldsymbol{k}$ implicitly maps the data to a Hilbert space $\mathcal{H}$, where the dot product is given by the kernel $\boldsymbol{k}$. Any algorithm that only depends upon the dot product between the data points can be made non-linear by employing the "kernel trick." Wherever a dot product is used, it is replaced with the kernel function. This is equivalent to first mapping the data points to the Hilbert space $\mathcal{H}$ and then carrying out the original algorithm. However, due to the use of the kernel, the mappings are not explicitly computed.

Figure 4 shows a simple binary classification example [87]. It is assumed that the true decision boundary is given by the ellipse in the input space. The points in the input space, $u = [u_1, u_2]^T$, are mapped to $\mathbb{R}^3$ using the mapping $\phi(u) = [u_1^2, \sqrt{(}2)u_1u_2, u_2^2]^T$. In $\mathbb{R}^3$, the decision boundary is transformed from an ellipse to a hyperplane, i.e. from a non-linear boundary to a linear one. There are many ways to carry out the mapping ϕ, but the above defined mapping has the important property that the dot product in the mapped space is given by the square of the dot product in the input space. This means that the dot product

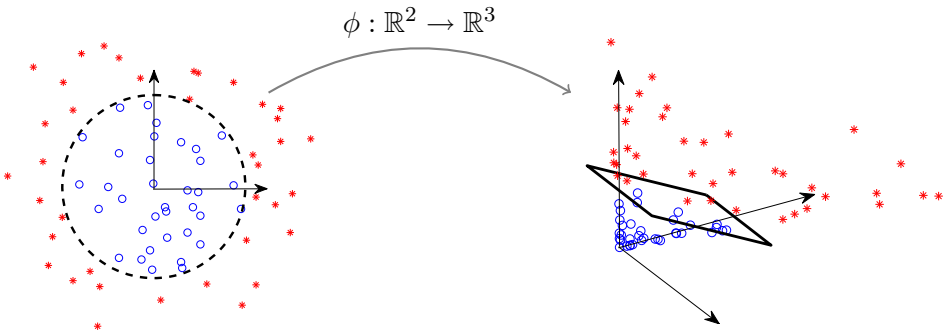


Figure 4: Toy example: Dot product in the mapped space can be computed using the kernel in the input space.

in the mapped space can be obtained without explicitly computing the mapping ϕ.

$$\begin{aligned}\phi(u) \cdot \phi(v) &= u_1^2 v_1^2 + 2u_1 v_1 u_2 v_2 + u_2^2 v_2^2 \\ &= (u_1 v_1 + u_2 v_2)^2 \\ &= (u \cdot v)^2 \\ &= \boldsymbol{k}(u, v).\end{aligned}$$

A example Mercer kernel is the Gaussian kernel,

$$\boldsymbol{k}(u_i, u_j) = \exp\left(-\frac{1}{2}(u_i - u_j)^T \Sigma^{-1}(u_i - u_j)\right), \tag{4}$$

where Σ a $d \times d$ symmetric, positive definite matrix. The Gaussian kernel will be used in the remainder of the thesis.

2.2 Kernel Principal Component Analysis

Kernel principal component analysis (**KPCA**) [86] is a non-linear extension of principal component analysis using a Mercer kernel $\boldsymbol{k}$. Given a set of n observations $\{u_i\}_{i=1}^n, u_i \in \mathbb{R}^d$ and a continuous, symmetric and positive definite function $\boldsymbol{k} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$, KPCA implicitly maps the data into a high-dimensional Hilbert space $\mathcal{H}$, using a non-linear mapping $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$. KPCA diagonalizes the covariance matrix

$$C_{\mathcal{H}} = \frac{1}{n}\sum_{i=1}^{n} \phi(u_i)\phi(u_i)^T,$$

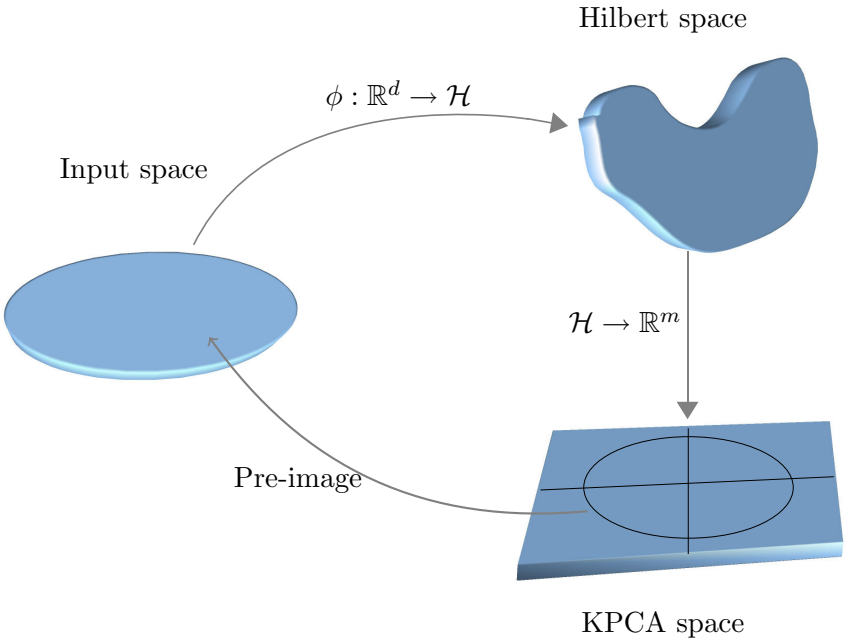


Figure 5: KPCA eigenspace representation. All points vectors in the input space are mapped nonlinearly to a Hilbert space where eigenvalue decomposition results in an m-dimensional KPCA space.

using the inner product matrix K, called the Gram/kernel matrix, with $K_{ij} = \phi(u_i)^T \phi(u_j) = \boldsymbol{k}(u_i, u_j)$. If $a_i^k, i = 1, \ldots, n$, and λ^k are the k-th eigenvector components and eigenvalue of the kernel matrix K, then the k-th eigenvector of the covariance matrix $C_{\mathcal{H}}$ is given by [59]

$$V^k = \frac{1}{\sqrt{\lambda^k}} \sum_{i=1}^{n} a_i^k \phi(u_i).$$

The embedding of the test point $\acute{u}$ in the KPCA space is obtained by projecting the test point onto each of the eigenvectors V_k. The projection on the k-th eigenvector is

$$f^k(\acute{u}) = V^k \cdot \phi(\acute{u}) = \frac{1}{\sqrt{\lambda^k}} \sum_{i=1}^{n} a_i^k \boldsymbol{k}(u_i, \acute{u}), \tag{5}$$

where $k = 1, \cdots, m$, and m is the total number of eigenvectors retained. For one of the training samples, the projection equation is equal to $f^k(u_i) = \sqrt{\lambda^k} a_i^k$. The extension of the embedding to new points defined by Equation (5) is know as the **out-of-sample** problem and it follows closely the Nyström approximation method [106, 107].

Another problem related to KPCA is the **pre-image** problem [4] (see Figure 5). The pre-image of $\psi \in \mathcal{H}$ is a point $\hat{u} \in \mathbb{R}^d$, such that $\phi(\hat{u}) = \psi$. ψ is known indirectly through

its projections, $\{f_\psi^k\}_{k=1}^m$, onto the KPCA eigenvectors, $\{V^k\}_{k=1}^m$. In general, the exact pre-image might not exist. Therefore, the pre-image methods are interested in finding the approximate pre-image satisfying the following optimality criteria:

$$\hat{u} = \arg\min_{\hat{u}} ||\phi(\hat{u}) - \psi||^2 .$$

It is assumed that ψ is a linear combination of training samples $\psi = \sum_{i=1}^n \eta_i \phi(u_i)$ where $\eta_i = \sum_k \frac{a_i^k}{\sqrt{\lambda^k}} f_\psi^k$. Assuming $\boldsymbol{k}$ is a normalized kernel,i.e. $\boldsymbol{k}(u,u) = 1$, the optimal $\hat{u}$ is given by [86]

$$\hat{u} = \frac{\sum_{i=1}^n \eta_i \boldsymbol{k}(u_i, \hat{u}) u_i}{\sum_{i=1}^n \eta_i \boldsymbol{k}(u_i, \hat{u})}, \tag{6}$$

This leads to the fixed point iteration scheme [86], which is susceptible to local minima and sensitive to initialization. Rathi et al [75] and Arias et al [4] propose an alternate to the fixed point iteration scheme. In the alternate methods, the kernel values $\boldsymbol{k}_i = \boldsymbol{k}(u_i, \hat{u})$ are approximated and placed back in Equation (6) to give a direct one step solution. In [75], the kernel values are approximated by using the relation between the kernel and the distance of the points in the Hilbert space $\mathcal{H}$ [56]

$$d_{\mathcal{H}}(\phi(\hat{u}), \phi(u_i))^2 = \boldsymbol{k}(\hat{u}, \hat{u}) + \boldsymbol{k}(u_i, u_i) - 2\boldsymbol{k}(u_i, \hat{u}).$$

For a normalized kernel this leads to

$$\boldsymbol{k}(u_i, \hat{u}) = \frac{1}{2}(2 - d_{\mathcal{H}}(\phi(\hat{u}), \phi(u_i))^2).$$

In [4], the kernel values are obtained by solving Equation (5) for $\boldsymbol{k}$. Given the projections f_ψ^k, the kernel values are obtained by solving a least square problem, and the solution is given by

$$\boldsymbol{k}_i = \sum_k a_i^k \sqrt{\lambda^k} f_\psi^k = \sum_k f^k(u_i) f_\psi^k. \tag{7}$$

The pre-image $\hat{u}$ is obtained by using Equation (7) in Equation (6). To summarize, the out-of-sample extension is to find the embedding/projections, $\{f^k(\acute{u})\}_{k=1}^m$, given a test point $\acute{u} \in \mathbb{R}^d$. The pre-image method is to find a point $\hat{u} \in \mathbb{R}^d$, given the embedding/projections $\{f_\psi^k\}_{k=1}^m$, in the KCPA space.

2.3 Contributions

KPCA is related to other manifold learning and dimensional reduction methods [15, 16, 48], such as multi-dimensional scaling, spectral clustering, locally linearly embedding. In manifold learning algorithms, the mapping in the embedding space is known only for the training examples, i.e, the out-of-sample and the pre-image methods do not exist [15]. However, in [15] the mappings are extended to new test points (out-of-sample extension), using the Nyström approximation method by carefully constructing the kernel matrix, K, for each method. In the remainder of this chapter, the procedure to carry out KPCA on kernel matrices, constructed for each method, is briefly explained, followed by the out-of-sample and pre-image problem. The contribution of the chapter is in providing the pre-image methods for other dimensional reduction and manifold learning techniques. This allows these techniques to be used for applications such as image de-noising, reconstruction and visualization, where the pre-image of the test point is required. This chapter concludes by practical applications of the proposed pre-image framework.

2.4 Multi-dimensional Scaling

In multi-dimensional scaling (**MDS**) the square of distance is computed between each pair of training examples [29], giving rise to the affinity matrix D. The matrix is doubly centered

$$\hat{D} = -\frac{1}{2} H D H,$$

by the centering matrix $H = (I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)$, where $\mathbf{1}$ is an $n \times 1$ column matrix of 1's. The centered distance matrix $\hat{D}$ plays the role of the Gram matrix K in the KPCA procedure. The embedding of a test point $\acute{u}$ is then given by Equation (5), with

$$K_{\acute{u}} = -\frac{1}{2} H (D_{\acute{u}} - \frac{1}{n} \hat{D}\, \mathbf{1}), \tag{8}$$

with $K_{\acute{u}} = [\boldsymbol{k}(u_1, \acute{u}), \cdots, \boldsymbol{k}(u_n, \acute{u})]^T$ and $D_{\acute{u}} = [d(u_1, \acute{u}), \cdots, d(u_n, \acute{u})]^T$, where $d(u_i, \acute{u})$ is the Euclidean distance between u_i and $\acute{u}$. The vector $K_{\acute{u}}$ defines the kernel evaluations necessary to evaluate Equation (5). To compute the pre-image, given the projections f_ψ^k, the kernel values $\boldsymbol{k}_i$ are computed using Equation (7). In case of MDS, the kernel $\boldsymbol{k}$ that

produced the matrix $\hat{D}$ is not normalized, i.e, $\boldsymbol{k}(\acute{u}, \acute{u}) \neq 1$. Therefore, Equation (6) can not be used to find the pre-image. Instead, the following equation is optimized to find the pre-image $\hat{u}$ [4]:

$$\hat{u} = \underset{\hat{u} \in \mathbb{R}^d}{\arg\min} \left(\sum_{i=1}^{n} \boldsymbol{k}(u_i, \hat{u}) - \boldsymbol{k}_i \right), \tag{9}$$

2.5 Isomap

Isomap [95] generalizes MDS by constructing a symmetric adjacency graph. Each point is connected to its neighbor with an edge weighted by the Euclidean distance. Dijkstra's algorithm is then used to find the shortest distance between each pair of data points. The isomap procedure approximates geodesic distances on the learned manifold. The pair-wise shortest distances form the matrix S. When the matrix S is doubly centered to give $\hat{S}$, $\hat{S}$ replaces the Gram matrix K in the KPCA procedure. To find the out-of-sample extension and the pre-image, Equations (8) and (9) are used with $\hat{S}$ instead of $\hat{D}$.

2.6 Spectral Clustering and Diffusion Maps

Spectral clustering (**SC**) [67] and diffusion maps (**DMAP**) [57] are clustering methods that cluster points using eigenvectors of matrices derived from the data. The matrix K is formed using a kernel such as the Gaussian kernel. The matrix is then normalized. In case of spectral clustering, the normalized Gram matrix is:

$$K_{sc} = D^{-\frac{1}{2}} K D^{-\frac{1}{2}}, \tag{10}$$

where D is a diagonal matrix each entry D_{ii} equal to the sum of the i-th row of matrix K.

In case of diffusion map, a two-step procedure occurs, where the following matrix is used in normalization step of Equation (10).

$$\hat{K} = P^{-1} K P^{-1},$$

with P a diagonal matrix whose entry P_{ii} equals the sum of i-th row of the matrix K. This initial normalization step separates the geometry of the space from the distribution of the points [57]. The kernel matrix for diffusion map is then given by

$$K_{dmap} = D^{-\frac{1}{2}} \hat{K} D^{-\frac{1}{2}},$$

where D is a diagonal matrix with D_{ii} equal to the sum of the i-th row of matrix $\hat{K}$. These normalized matrices are then used in the KPCA procedure. In particular, the out-of-sample and the pre-image are obtained using Equation (5) and Equation (6) respectively.

2.7 Locally Linear Embedding

The locally linear embedding (**LLE**) [80] method assumes that the data lies on a non-linear manifold, which can be linearly approximated locally. LLE builds a weight matrix W, whose i-th row contains the linear coefficients that sum to unity and reconstruct the data point u_i from its p neighbors. The weights are found by minimizing the following equation:

$$\left(\sum_j W_{ij} u_j - u_i \right)^2$$

Define a matrix M

$$M = (I - W)^T (I - W)$$

The embedding for LLE is obtained from the smallest eigenvectors of M. The leading eigenvectors of the kernel matrix given by

$$K_{lle} = \mu I - M,$$

are the smallest eigenvectors of M [48]. Therefore, the kernel matrix K_{lle} can be used in the KPCA procedure. Equation (5) can not be used to extend the embedding to new data points, since the kernel $\boldsymbol{k}$, which creates the kernel matrix K_{lle}, is not known analytically. Instead the following equation is used to find the embedding of the new points [15, 82]:

$$f^k(\acute{u}) = \sum_{i=1}^{n} f^k(u_i) w_i, \tag{11}$$

where w_i is the weight of u_i in the reconstruction of $\acute{u}$. If u_i is not a neighbor of $\acute{u}$, then $w_i = 0$. To find the pre-image, given the embedding $f^k(\hat{u})$, the weights w_i are obtained by solving Equation (11) for w_i.

$$w_i = \sum_k f^k(u_i) f^k_{\psi}. \tag{12}$$

The pre-image is then given by

$$\hat{u} = \sum_{i=1}^{n} w_i u_i. \tag{13}$$

2.8 Kernelized Locally Linear Embedding

Kernelized locally linear embedding (**KLLE**) [36] generalizes the LLE algorithm to non-linear manifolds by employing the kernel trick. A Mercer kernel $\boldsymbol{k}$ implicitly maps the data points to a higher dimensional Hilbert space $\mathcal{H}$, where the inner product is given by the kernel. The distance between the points are computed in the feature space and (assuming a Gaussian kernel) are given by [83]

$$d_{\mathcal{H}}(\phi(u_i), \phi(u_j)) = 2(1 - \boldsymbol{k}(u_i, u_j)).$$

The weight matrix W is constructed, whose i-th row contains the linear coefficients that sum to unity and reconstruct the Hilbert space data point $\phi(u_i)$ from its p neighbors. The weights are found by minimizing the following equation:

$$\left(\sum_j W_{ij}\phi(u_j) - \phi(u_i) \right)^2 . \tag{14}$$

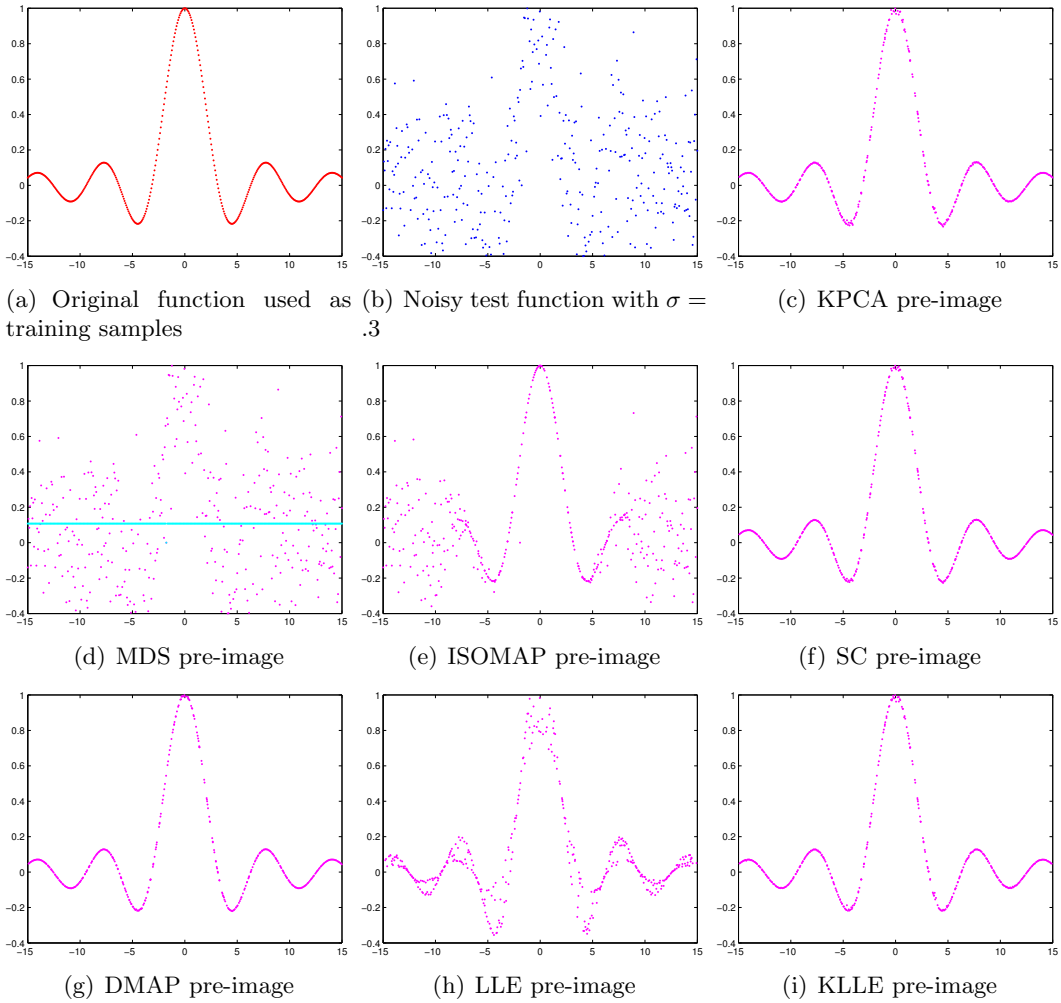
A matrix $M = (I - W)^T(I - W)$ is constructed, whose smallest eigenvectors give the embedding for the KLLE. Similar to the procedure for LLE, a matrix

$$K_{klle} = \mu I - M$$

is defined and used in the KPCA procedure. The projection of a test point $\acute{u}$ can be computed using the Equation (11). However, the weights w_i in Equation (11) are computed by minimizing Equation (14). The pre-image of $\psi \in \mathcal{H}$, which is known by its embedding f_ψ^k, is obtained by first estimating the weights w_i (Equation (12)). In case of LLE the weights w_i are then used to obtain the pre-image (Equation (13)). This is not possible for KLLE, since the weights reconstruct the point in the Hilbert space $\psi = \sum_{i=1}^n w_i\phi(u_i)$. We propose to use Equation (6) and Equation (7) to compute the pre-image.

$$\hat{u} = \frac{\sum_{i=1}^n w_i w_i u_i}{\sum_{i=1}^n w_i w_i}, \tag{15}$$

where the first w_i comes from the fact that w_i's are the coefficients for the reconstruction of the test point in Hilbert space, and the second w_i comes from the fact that w_i is also the inner product of the embedding in the Hilbert space (Equation (12)).



(a) Original function used as training samples (b) Noisy test function with $\sigma = .3$ (c) KPCA pre-image

(d) MDS pre-image (e) ISOMAP pre-image (f) SC pre-image

(g) DMAP pre-image (h) LLE pre-image (i) KLLE pre-image

Figure 6: Pre-image computation for different methods

A pre-image method for KLLE is described in [74], where the point in the input space corresponding to the projection f_ψ^k must be known, for the pre-image method to work. However, in the method proposed here, the pre-image can be calculated given only f_ψ^k. This may be useful in cases where, for example, the pre-image of the mean of two projections is needed, and for which a corresponding input point may not exist.

2.9 Experiments

Toy example: The pre-image framework presented in this chapter is first tested on a toy example. Consider the function $y = \text{sinc}(x)/x$. A total of 400 points are uniformly sampled over the domain $x = [-10, 10]$ as shown in Figure 6(a). These points are taken as training data with which the kernel matrix is formed for each of the methods. The test points are

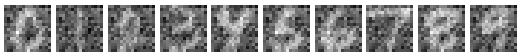
Table 1: Average PSNR(db) over ten digits

noise level	KPCA	SC	DMAP	LLE	KLLE	PCA
$\sigma = .2$	25.30	25.31	25.31	16.72	18.91	19.84
$\sigma = .5$	21.64	21.64	21.64	16.67	18.84	12.10
$\sigma = 1$	17.87	17.78	17.72	17.01	18.5	12.11

generated by corrupting the function with Gaussian noise of $\sigma = .3$, as shown in Figure 6(b). The pre-images are computed for each method and are shown in Figures 6(c)-6(i). Since the data is two dimensional, the eigenvalue decomposition for the kernel matrix for MDS K_{mds} can have a maximum of two eigenvectors. Retaining only one eigenvector results in the cyan line in Figure 6(d), whereas retaining two eigenvectors reconstructs the original noisy points. This shows the equivalence of MDS to principal component analysis PCA. Isomap kernel K_{isomap} learns the geodesic distance on the manifold, and is, therefore, better than MDS in reconstructing the true function. KPCA, SC and DMAP use different normalization of the kernel matrix, but produce quite similar results. Similarly, the kernelized version of LLE performs better than the plain LLE.

De-noising: An application where pre-image framework can be used is image de-noising. USPS database of handwritten digits [85] is used for the experiment. 30 images of each of the ten digits are taken as training samples to form the kernel matrix K, on which KPCA is performed. Given a noisy image, de-noised image is obtained by computing the pre-image using each of the above mentioned methods. The results are shown in Table 1, where the average PSNR is computed over all digits for different level of Gaussian noise. Figures 7 and 8 show the results for the case of Gaussian noise level $\sigma = .5$ and $\sigma = 1$.

(a) Test digits



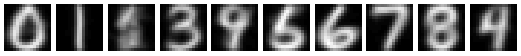
(b) Test digits corrupted by Gaussian noise of $\sigma = .5$

(c) KPCA De-noising

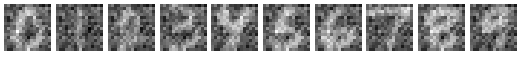
(d) SC De-noising

(e) DMAP De-noising

(f) LLE De-noising



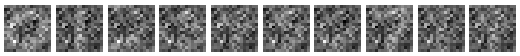
(g) KLLE De-noising



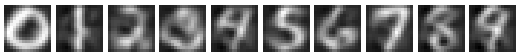
(h) PCA De-noising

Figure 7: Image de-noising, Gaussian noise level $\sigma = .5$

(a) Test digits



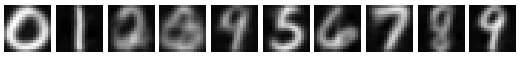
(b) Test digits corrupted by Gaussian noise of $\sigma = 1$



(c) KPCA de-noising

(d) SC de-noising

(e) DMAP de-noising

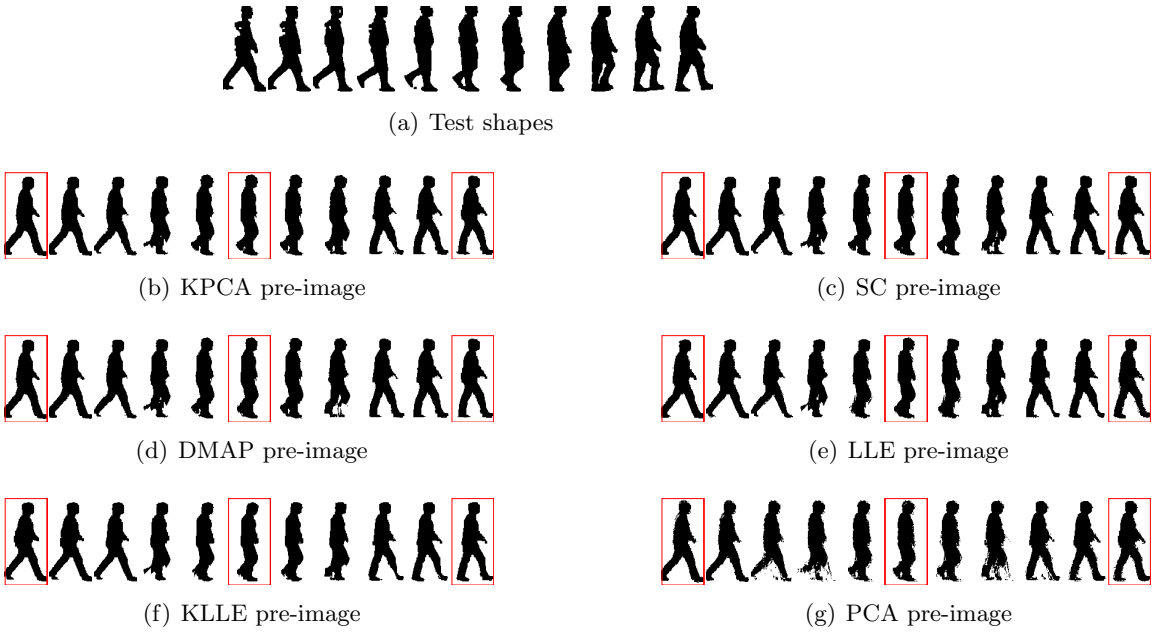


(f) LLE de-noising

(g) KLLE de-noising

(h) PCA de-noising

Figure 8: Image de-noising, Gaussian noise level $\sigma = 1$



(a) Test shapes

(b) KPCA pre-image

(c) SC pre-image

(d) DMAP pre-image

(e) LLE pre-image

(f) KLLE pre-image

(g) PCA pre-image

Figure 9: Pre-image computations. The pre-images without the red bounding box are the ones whose embedding was approximated using interpolation in the embedding space.

Interpolation in the Hilbert space: In this experiment, 68 silhouettes of a walking person are taken as training shapes and learned. Given a sequence of test silhouettes, some of the intermediate ones are discarded. The rest are mapped to the learned space, where linear interpolation is carried out to get the approximate mapping for the discarded shapes. The pre-image of the mappings are then carried out. The results are shown in Figure 9, where the first row shows the test silhouettes. The other rows show the pre-image. The pre-images without the red bounding box are the ones whose embeddings were approximated using interpolation in the embedding space.

CHAPTER III

KPCA-BASED EIGENSPACE REPRESENTATION FOR TRACKING

3.1 Introduction

The chapter presents a non-rigid object localization and segmentation algorithm using a KPCA-based eigenspace representation. Localization and segmentation are carried out by deriving a similarity function in the KPCA eigenspace. The KPCA space is related non-linearly to the input space, which results in a non-linear algorithm. The de-noising and the clustering capabilities of the KPCA procedure lead to a localization procedure that is robust to noise and occlusions. A unique feature of the approach is that it permits segmentation in addition to localization, when multiple templates of the target are given.

3.2 Related Work

Localization methods determine the correspondence of the object region in consecutive images by using a template or a set of templates of the target object to define a region descriptor, which provides characteristic information about the object. Target localization is then formulated as a region descriptor search scheme using a pre-determined similarity function.

Histogram-based: Comaniciu et al [28] use a histogram weighted by a spatial kernel as a probability density function of the object region. The correspondence of the target object between sequential frames is established at the region level by optimizing the Bhattacharya coefficient between the target and the candidate distributions using mean-shift [26]. Histograms discard spatial information, which becomes problematic when faced with occlusions and/or the presence of target features in the background. Attempts to incorporate spatial information into the descriptor include [47, 113, 19]. Instead of using the Bhattacharya coefficient, [47] use the Matusita distance between kernel-modulated histograms. The Matusita distance is optimized using Newton-style iterations, which provides faster convergence than

the mean-shift. Additionally, limitations of using a single kernel are provided and an extension using multiple spatially distributed kernels is developed. Multiple kernels are chosen so as to increase the rank of the resulting system. Another extension to multiple-kernel tracking is [113], where multiple collaborative kernels are placed over the target object. In [19], histograms were generalized to include spatial information, leading to spatiograms. A spatiogram augments each histogram bin with the spatial means and covariances of the pixels comprising the bin.

Nonparmatric kernel-based: The algorithms defined above require computing the probability density functions (histograms/spatiograms), which becomes computationally expensive for higher dimensions. An additional problem associated with computing probability density functions is the sparseness of the observations within the feature space, especially when the sample set size is small. The sparseness makes similarity measures, such as Bhattacharya coefficient, computationally unstable [108]. Methods, such as [38, 91, 108] define similarity functions between kernel density estimates of the template and target distribution in a joint feature-spatial space. The relationship between the appearance and spatial information is more fully exploited as compared to the previously discussed approaches. Since these methods employ non-parametric density comparison techniques, the intermediate step of estimating the density function is not carried out and the similarity functions are defined directly on the samples obtained from the target and the candidate regions.

Statistical eigenspace-based: An alternate image region descriptor is that of the covariance matrix associated to the spatial and the appearance information of a target object [73]. In [73], exhaustive search is performed over the image domain to find the region whose covariance matrix has the smallest distance to the model covariance matrix. Due to its global nature, the algorithm can recover from total occlusions or large movements. However, it is also susceptible to noise, background clutter, and false positives. Karasev et al [51] extend the covariance tracker by considering spatially weighted mean and covariance descriptions of the target object, and provide a variational approach to target localization.

Avidan [14] treats tracking as a classification problem and trains several weak binary

classifiers to separate pixels that belong to the object from the pixels that belong to the background. A single strong classifier is tested on all pixels in the current image to create a confidence map. Mean shift is run on the confidence map to find the object rectangle.

Essential to improved tracking is the derivation of a model that can capture the relationship between the purely image-based observations and the spatial content associated to said observations. Unsupervised learning techniques such as principal component analysis **(PCA)** have been used to measure the correlation among the pixels of the templates of the target. The templates are vectorized to form a matrix $\mathbb{D} = [I_1, \cdots, I_N]$, where each column I_i is a vectorized template. The covariance matrix obtained from the data in $\mathbb{D}$ is diagonalized to obtain a low dimensional eigenspace representation of the target. This representation has been used for tracking in [21] and [62]. In [62], the subspace is also incrementally updated to account for appearance and illumination change. Chin et al [27] use nonlinear subspace representation derived using kernel methods [86] for face recognition and visual tracking. Tsai et al. [98] perform PCA on a collection of vectorized signed distance maps of the training shapes to incorporate the shape model into the segmentation procedure. This fundamental concept has been applied to tracking in [31, 32, 64]. In these settings, each vectorized template I_i is an observation with the implicit assumption that the image appearance will remain similar to the training templates. However, under partial or extensive occlusions, this assumption will not hold and the tracker may give erroneous results.

Contribution: This chapter connects nonparametric, kernel-based methods with statistical eigenspace methods to derive a target localization strategy. Each feature vector associated to a pixel of the target object describes an observation of the target object, whose overall joint appearance-spatial geometry is learned using the eigenspace representation associated to the collection of feature vectors forming the target. Kernel principal component analysis (**KPCA**) is used for the eigenspace representation. KPCA is attractive because of its nonlinear nature and noise suppression characteristics. The eigenspace representation provides a compact and robust description of the target being tracked. Thus, while tracking is performed through pixel-wise computations in the eigenspace, the pixels

are implicitly connected by the eigenspace representation. A unique feature of the approach is that it permits segmentation in addition to localization.

3.3 KPCA-based Eigenspace Representation of the Target

3.3.1 Extracting Target Feature Vectors

The feature vector associated to a given pixel is a d-dimensional concatenation of a p-dimensional appearance vector and a 2-dimensional spatial vector $u = [\mathcal{F}(x), x]$, where $\mathcal{F}(x)$ is the p-dimensional appearance vector extracted from $\mathcal{I}$ at the spatial location x,

$$\mathcal{F}(x) = \Gamma(\mathcal{I}, x),$$

where Γ can be any mapping such as color $\mathcal{I}(x)$, image gradient, edge, texture, etc., any combination of these, or the output from a filter bank (Gabor filter, wavelet, etc.).

The feature vectors are extracted from the segmented target template image(s). The set of all feature vectors define the target input space $\mathbb{D}$,

$$\mathbb{D} = \{u_1, u_2, ..., u_n\},$$

where n is the total number of feature vectors extracted from the template image(s).

3.3.2 Target Eigenspace Representation

The eigenspace representation of the input space comes from KPCA, which maps the data into a high-dimensional Hilbert space $\mathcal{H}$, using a nonlinear mapping $\phi : \mathbb{R}^d \rightarrow \mathcal{H}$ and diagonalizing the covariance matrix $C_{\mathcal{H}}$, using the kernel trick. Following the procedure in Chapter 2, the eigenvectors of covariance matrix $C_{\mathcal{H}}$ are given by

$$V_k = \frac{1}{\sqrt{\lambda^k}} \sum_{i=1}^{n} a_i^k \phi(u_i),$$

where $a_i^k, i = 1, \ldots, n$ and λ^k are the k-th eigenvector and eigenvalue of the Gram matrix, K, with $K_{ij} = \phi(u_i)^T \phi(u_j) = \boldsymbol{k}(u_i, u_j)$. A test point u is represented in the KPCA space by projecting it onto the eigenvectors V^k. The projection on the k-th eigenvector is

$$f^k(u) = V^k \cdot \phi(u) = \frac{1}{\sqrt{\lambda^k}} \sum_{i=1}^{n} a_i^k \boldsymbol{k}(u_i, u). \tag{16}$$

In the KPCA space, a feature point u is represented as $\boldsymbol{f(u)} = [f^1(u), \ldots, f^m(u)]^T$, where m is the total number of eigenvectors retained. All further computations are performed in the KPCA space. Since the KPCA space is related nonlinearly to the input space, the resulting algorithm is nonlinear.

The projection Equation (16) is computed in the high-dimensional space through the kernel in terms of linear combinations of all the input vectors $\{u_i\}_{i=1}^n$. If the total number of elements n is large, Equation (16) becomes unsuitable for on-line tracking applications. The projection equation can be approximated using fewer samples $c_i^k \in \mathbb{R}^d$ with coefficients w_i^k as

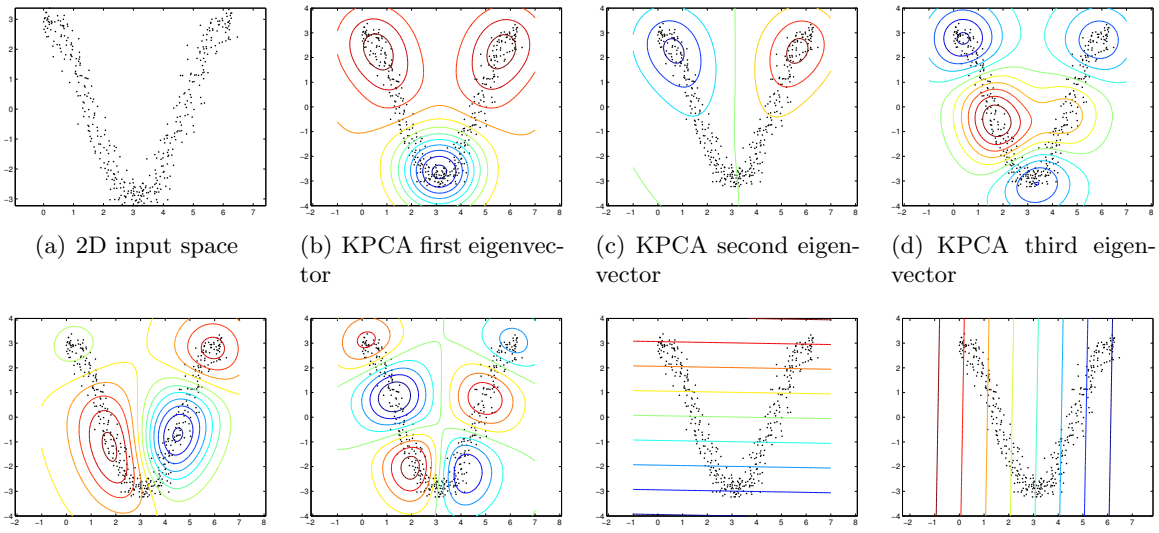
$$f^{k*}(u) = \sum_{i=1}^{l} \gamma_i^k \boldsymbol{k}(c_i^k, u), \tag{17}$$

where $l << n$ The points c_i^k and their coefficients w_i^k are found using the reduced kernel representation (see Chapter 4).

3.3.3 Properties of the Eigenspace Representation

The KPCA-based eigenspace representation has several properties that are advantageous when tracking. In particular, the benefits described below will transfer to the similarity function to be defined shortly.

Enhanced clustering/discrimination: Unsupervised clustering algorithms such as PCA and KPCA can capture a number of clusters/features equal to the number of eigenvectors retained plus one [37]. In case of PCA, the maximum eigenvectors are limited by the the dimension of the input space, $\mathbb{R}^d$. For KPCA, the maximum number of eigenvectors is given by the number of data points in the input space, n. Given that $n >> d$, KPCA can represent significantly more clusters [37], thus its representation capacity is richer. Figure 10 depicts isoclines of the eigenvector coefficients for the eigenvectors learned in case of a two-dimensional data of Figure 10(a). PCA has a maximum of two eigenvectors. The eigenvectors learned using KPCA capture different aspects of the two-dimensional dataset as shown in Figures 10(b)-10(f). The eigenvectors capture non-linear relations among the dataset. These properties can be very useful for developing an object tracking algorithm, where the relationship among three or more pixels provide useful information about the



(a) 2D input space (b) KPCA first eigenvector (c) KPCA second eigenvector (d) KPCA third eigenvector

(e) KPCA fourth eigenvector (f) KPCA fifth eigenvector (g) PCA first eigenvector (h) PCA second eigenvector

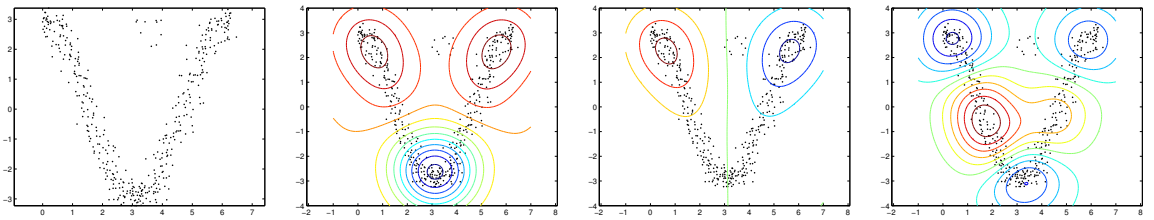
Figure 10: KPCA vs PCA: KPCA has a larger theoretical number of retainable eigenvectors versus PCA, therefore KPCA can capture more clusters/features.

target [66].

Noise/outlier removal: There are two ways noise and outliers can effect the tracking process, either during the learning phase (the noise present in the sample set used for training) or during the tracking phase. The use of KPCA helps in reducing the effect of noise and outliers in both cases. Noise suppression is achieved by selecting less than the maximal eigenvalues possible. For example, Figure 11(a) shows a case with outliers added to the training set. The isoclines of the leading eigenvector coefficients remain the same as the outlier free example depicted in Figure 10. Figure 11 shows the reconstruction of the samples using two, four, eight and ten eigenvectors. The reconstruction shows that the eighth eigenvector and beyond will capture noise and therefore can be ignored. In the next section, we will show that the projections on to the eigenspace go to zero when data is far from the template samples. This helps in reducing the effect of noise/outliers during the tracking phase. The remaining sections exploit these properties for tracking.

3.4 Similarity Measure in KPCA Space

Object tracking in the KPCA space requires defining a similarity function in the KPCA space. Here, the similarity function will measure the similarity of a feature vector to the

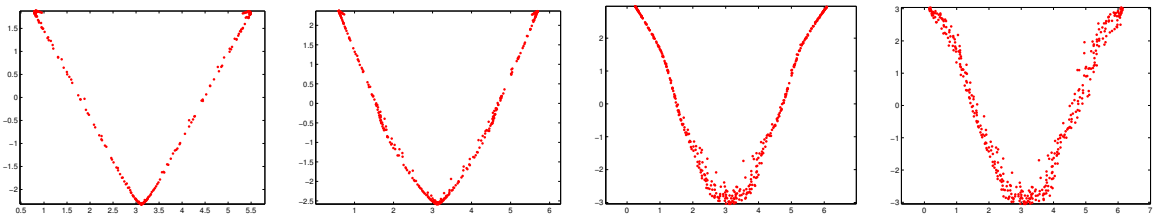


(a) Input space with outliers

(b) KPCA first eigenvector

(c) KPCA second eigenvector

(d) KPCA third eigenvector



(e) Reconstruction w/two eigenvectors

(f) Reconstruction w/four eigenvectors

(g) Reconstruction w/eight eigenvectors

(h) Reconstruction w/ten eigenvectors

Figure 11: Noise/outlier rejection: Top row shows that the projections onto the leading eigenvectors remain the same as in Figure 10. The reconstructions in the bottom row show that the eighth and beyond eigenvectors capture noise.

learned model. As per Section 3.3, the similarity function defined in the eigenspace will capture nonlinear relationships between the feature vectors.

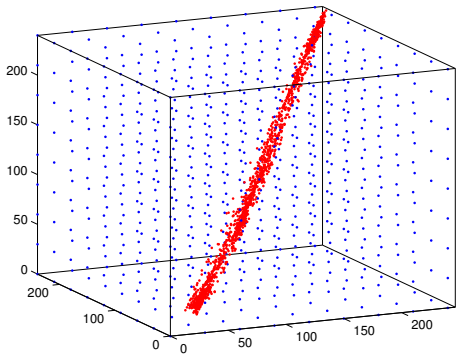
3.4.1 Feature Vector Similarity Function

The similarity function comparing a given feature vector (or set of feature vectors) to the learned model will exploit the geometric properties of the KPCA eigenspace. Under Gaussian kernel, the KPCA space is a high-dimensional elliptical space. A feature vector u is represented by $\boldsymbol{f(u)} = [f^1(u), \ldots, f^m(u)]^T$, where each $f^k(u)$ is computed using Equation (16). It is evident from Equation (16) that the function $f^k(u)$ tends to zero as the vector u recedes from the input space $\mathbb{D}$. Similarly, when u is a feature vector representative of an input space element, the distance of the vector $\boldsymbol{f}(u)$ from the KPCA-space origin, is bounded and given by

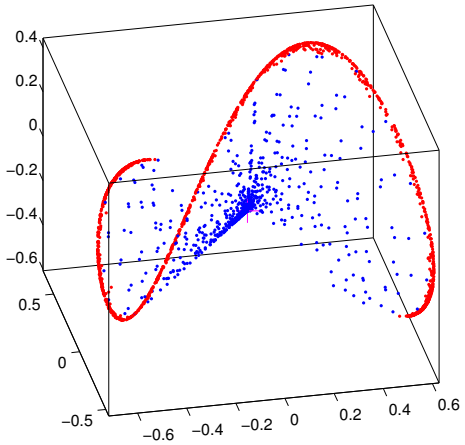
$$||\boldsymbol{f}(u)|| = \langle \boldsymbol{f}(u), \boldsymbol{f}(u) \rangle^{\frac{1}{2}} = 1.$$

However, since the projection $f^k(u)$ is approximated using the Nyström approximation method [106], the distance computation satisfies the inequality $||\boldsymbol{f}(u)|| \leq 1$ [4].

The net result is to map the collection of data into a high-dimensional elliptical space.



(a) Input space. Target color values are shown in red.



(b) KPCA space. KPCA space is bounded by the target object values.

Figure 12: The target object color values lie on the surface of the ellipse in KPCA space, while other color values lie interior to the ellipse. The square distance from the origin can be used as a similarity function.

An element that coincides or agrees with the data will map onto the surface of the hyper ellipse, whereas one that does not will map close to the origin based on the degree to which it is an outlier. Hence, the squared distance from the origin operates like similarity measure [99], with the further property that it is functionally similar to a likelihood. The similarity of a test feature vector u to the target model $\mathbb{D}$ is defined as

$$\mathcal{J}(u) = ||\boldsymbol{f}(u)||^2 = \sum_{k=1}^{m} \left(f^k(u) \right)^2, \tag{18}$$

where m is the total number of eigenvectors retained. In Figure 12, the target color values shown in red are learned using KPCA, while the blue points depict other points in the domain. The learned KPCA space is shown in Figure 12(b), for which both the red and blue points are mapped. The target object color values lie on the surface of the ellipse in KPCA space, while all other color values lie within the ellipse. The square distance from the origin (Equation (18)) is used as a similarity function. Importantly, the similarity function rewards inliers and ignores outliers, thus lending robustness to outliers (such as target occlusions and background clutter). Further details on the statistical interpretation of this approach and its relation to robust density matching can be found in Chapter 5.

3.4.2 Region similarity measure

To measure the similarity of a region $\mathcal{R}$, all feature vectors u falling within the region $\mathcal{R}$ are extracted. The center of the region is taken to be the origin of the spatial domain. Performing the similarity calculation, Equation (18), for all the vectors in the region $\mathcal{R}$ then taking the sum results in a measure of how well, as a whole, the region $\mathcal{R}$ represents the target model,

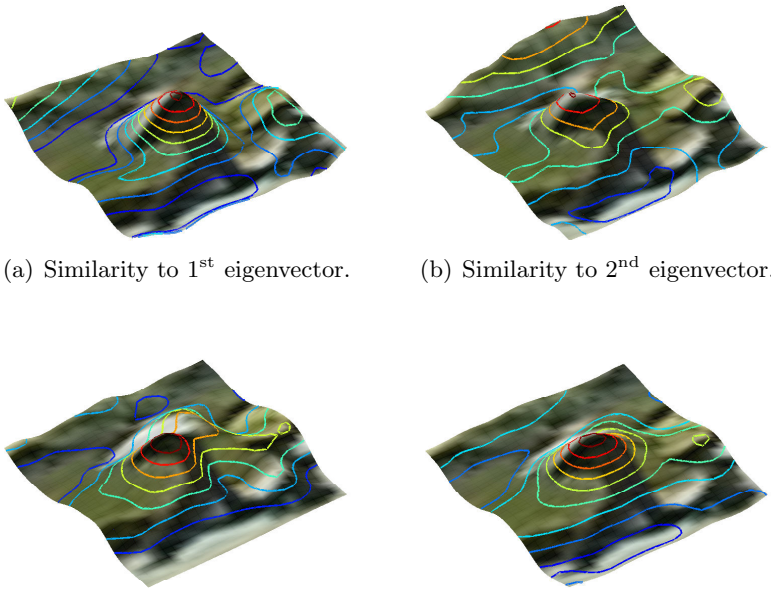
$$\mathcal{J}(\mathcal{R}) = \sum_{u \in \mathcal{R}} \sum_{k=1}^{m} \mathcal{J}^k(u) = \sum_{u \in \mathcal{R}} \sum_{k=1}^{m} \left(f^k(u) \right)^2 . \tag{19}$$

Localization will involve finding the center location of the region $\mathcal{R}$ with the maximum $\mathcal{J}(\mathcal{R})$ as this will correspond to the most likely location of the target. Figure 13 shows the computation of region similarity measure for the first three eigenvectors and total computation $J(\mathcal{R})$, which is equal to the sum of the three individual region similarity measures. In all the cases, the peak is found at the location of the target. This means that the target can be robustly located even under partial occlusions since the eigenvectors corresponding to visible parts are used to track the object.

3.4.3 Evaluation of Region Similarity Function on a 1D Synthetic Signal Detection

To demonstrate the properties of the KPCA eigenspace representation and its associated similarity measure (19), the method is applied to a 1D synthetic example as in [91]. Consider detecting a 1D template signal embedded in a random signal of thrice its length and corrupted by (1) Gaussian noise, (2) log-normal noise, or (3) occlusions. The corrupted signal is searched over the whole domain to find the template. Comparison algorithms include (1) kernel density correlation **(KDC)** [91], (2) covariance-based detection **(COV)** [73], and (3) sample correlation **(SC)**. KDC, which is based on kernel density estimation, is used for stereo registration and tracking [91]. KDC outperforms Mutual Information, a standard measure used for robust estimation [104].

Figure 14 depicts an instance of the random signal, in which the template signal is embedded starting at 247 with 40% occlusion and Gaussian noise. The original template signal is shown in red. For all experiments, the true location of the template signal in the



(a) Similarity to 1st eigenvector. (b) Similarity to 2nd eigenvector.

(c) Similarity to 3^{rd} eigenvector. (d) Total similarity.

Figure 13: Region similarity measure for the first three eigenvectors. For each eigenvector, the region similarity peaks at the location of the target object. A target can be robustly located even under occlusions, as the eigenvectors corresponding to visible portions will score higher and thus have more influence.

random signal is fixed at 247. The number of eigenvectors m used to compute the region similarity is eight. To build the vectors, the signal value and its spatial location is used. For each set of nuisance parameters (noise level and % occlusion), the experiment is repeated 200 times.

Detection performance is shown by plotting the detection probability (true positives) vs. the false alarm probability, as the discrimination threshold is varied. The plot, known as the receiver operative characteristic (**ROC**), is widely used to analyze detector performance. The closer the curve hugs the upper-left corner, the better the performance. In Experiment 1, the signal is corrupted at different i.i.d. zero-mean Gaussian noise levels. ROC curves for noise levels 30 and 60 are shown in 15(a) and 15(b). SC performs the best; under additive white Gaussian noise, SC is the minimum variance unbiased estimator. For Experiment 2, zero-mean log-normal noise is added to simulate background clutter. Results are depicted in Figures 15(c) and 15(d). SC is sensitive to outliers, so its performance deteriorates. The proposed measure and KDC are robust to outliers. Thirdly, performance under occlusion

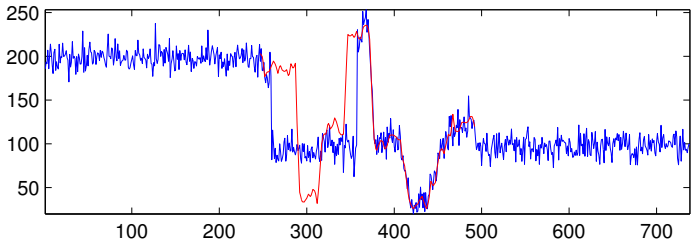
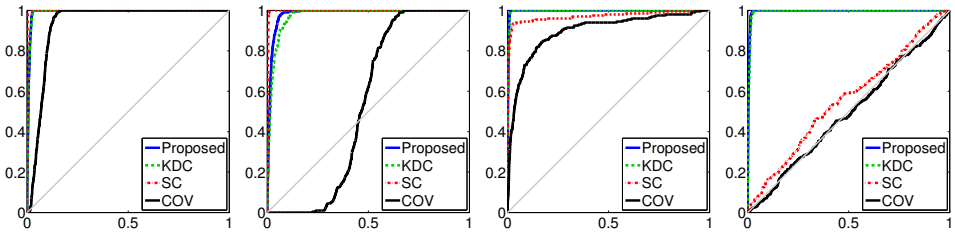
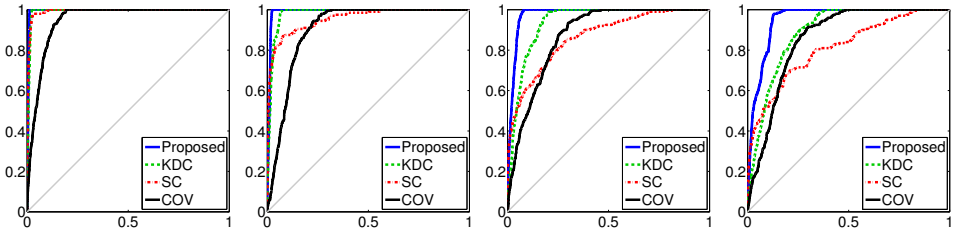


Figure 14: Template signal (red) overlayed on a sample corrupted signal (blue).



(a) Gaussian, $\sigma = 30$. (b) Gaussian, $\sigma = 60$. (c) Log-Normal, $\sigma = 2$. (d) Log-Normal, $\sigma = 4$.

Figure 15: ROC curves for Gaussian and Log-Normal noise.



(a) 10%. (b) 30%. (c) 50%. (d) 60%.

Figure 16: ROC curve for various occlusion levels.

is tested. The template signal is occluded at random locations for occlusions of 10% to 60%. The results, shown in Figure 16, clearly show the proposed measure's robustness to occlusions. As per Yang and Duraswami [108], density-based tracking using probabilistic similarity measures degrades with increasing data dimension. Therefore, the 1D results are the best that a given algorithm will perform at, since discrimination capability can only degrade for 2D and higher. The results show that the proposed method has much better ROC curves for noise and for occlusions. The same is expected for higher dimensional source data.

3.5 Object Tracking

While target detection can be performed exhaustively over the image, as was done with the 1D synthetic signal from Section 3.4.3, such a strategy can be computationally costly within the context of visual tracking. This section presents two variational strategies for locally optimizing the region similarity given an initial estimate of the optimal region.

3.5.1 Variational Target Localization

Assume that the target object undergoes a geometric transformation to a region $\tilde{\mathcal{R}}$, such that $\mathcal{R} = T(\tilde{\mathcal{R}}, a)$, where a is the transformation parameter. Let $\acute{u}$ be a feature vector sampled from the region $\tilde{\mathcal{R}}$ whereby $\acute{u} = [\mathcal{F}(\tilde{x}), T(\tilde{x}, a)]^T = [\mathcal{F}(\tilde{x}), x]^T$. This section derives a gradient descent procedure to maximize the region similarity (Equation (19)) with respect to the transformation parameter a. The gradient is

$$\nabla_a \mathcal{J}(\tilde{\mathcal{R}}) = \sum_{\acute{u} \in \tilde{\mathcal{R}}} \nabla_a \mathcal{J}(\acute{u}) = \sum_{\acute{u} \in \tilde{\mathcal{R}}} \sum_{k=1}^{m} 2 f^k(\acute{u}) \nabla_a f^k(\acute{u}),$$

where

$$\nabla_a f^k(\acute{u}) = \nabla_a T(\tilde{x}, a). \nabla_x f^k(\acute{u}),$$

with $\nabla_a T(\tilde{x}, a)$ a $g \times 2$ Jacobian matrix of T given by $\nabla_a T = [\frac{\partial T}{\partial a_1}, \ldots, \frac{\partial T}{\partial a_g}]^T$, where g is the number of transformation parameters. The gradient $\nabla_x f^k(\acute{u})$ is

$$\nabla_x f^k(\acute{u}) = \frac{1}{\sigma_s^2} \sum_{i=1}^{l} w_i^k \boldsymbol{k}(c_i^k, \acute{u},)(\pi_s(c_i^k) - x),$$

where π_s is a function that takes full d-dimensional vector and returns only the spatial values, and σ_s is the spatial bandwidth parameter of the kernel $\boldsymbol{k}$. The transformation parameters are then updated using the following equation:

$$a(t+1) = a(t) + \delta t \nabla_a \mathcal{J}(\tilde{\mathcal{R}}) \tag{20}$$

where m is the total number of eigenvectors used and δt is the time step. While gradient ascent is derived here, other variational optimization strategies can be applied.

An alternate update using mean-shift: In some instances, the mean shift method [26] can be used to find the target location. Consider the case of translational motion only.

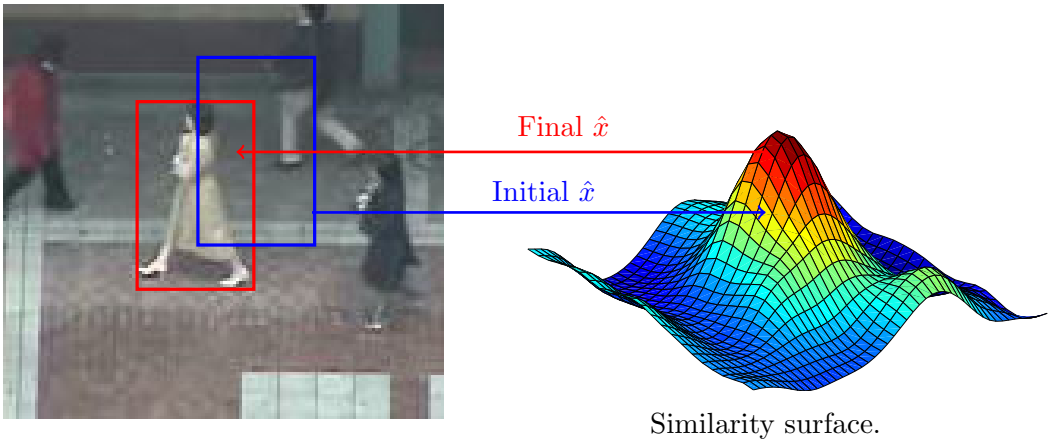


Figure 17: The similarity measure represents an unnormalized density estimate. Mean-shift can be used to find the mode of the density function.

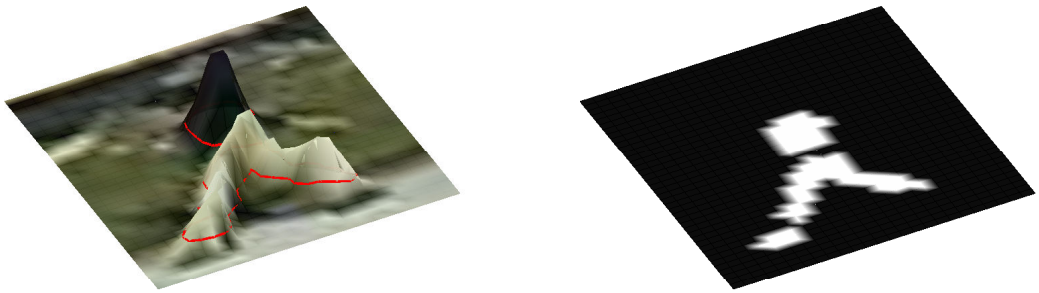
Assume $\hat{x}$ is the estimated location of the target object, and the region $\mathcal{R}_{\hat{x}}$ is centered at $\hat{x}$. Let the feature vector u be given by $u = [\mathcal{I}(x), x - \hat{x}]$. Due to the probabilistic properties of the similarity measure, discussed in Section 3.4.1, $\mathcal{J}(\mathcal{R}_{\hat{x}})$ represents an unnormalized density estimate computed at $\hat{x}$; see Figure 17. To find the local mode mean shift iterations can be used (see Appendix A). Mean shift is a non-parametric, iterative procedure for locating stationary points of a density function. The update location is

$$\hat{x} = \frac{\sum_{u \in \mathcal{R}_{\hat{x}}} \sum_{k=1}^{m} f^k(u) \sum_{i=1}^{l} w_i^k \boldsymbol{k}(c_i^k, u)(\pi_s(c_i^k) - x)}{\mathcal{J}(\mathcal{R}_{\hat{x}})}. \tag{21}$$

Iterating (21) finds the region whose similarity measure is the local density maximum.

3.5.2 Segmentation by Thresholding

Once the target has been localized, to perform segmentation, the similarity measure (Equation (18)) is computed for each feature vector falling within the region $\mathcal{R}$. As explained in Section 3.4.1, feature vectors similar to the learned model are mapped close to the surface of the KPCA space, while outlier feature vectors are mapped closer to the origin, depending upon the degree to which they are outliers. To obtain the segmentation, thresholding is performed on the squared distance from the origin in the KPCA to get a binary mask for the target object. In Figure 18(a), the image that falls within the region $\mathcal{R}$ is used to color the surface which is created as a result of measuring the similarity of each feature vector to



(a) Height indicates similarity to learned model

(b) Thresholding Result

Figure 18: Segmentation by thresholding

the learned model. The red contour in the figure represents the thresholding of the values to get the binary mask shown in Figure 18(b).

3.6 Experiments

The tracking performance of the proposed method is tested on a number of real world objects such as cars, a dog, a fish, walking and running people, and face tracking. Some videos are standard test sequences with scene clutter, noise, scale changes, and occlusions. In most, the target area is small compared to the image dimensions.

At its most complex, visual object tracking consists of target localization and segmentation. The results for target localization and segmentation are provided in Section 3.6.1, while in Section 3.6.2 only target localization is considered. For target localization and segmentation, three templates are used to learn the model. For target localization, only one template is used. Therefore, if only localization is needed, the initialization time is reduced and the target can start tracking given the location of the target in the first frame. In Section 3.6.3, tracking results on combined infrared sequences and color sequences are shown. Section 3.6.4 concludes with tracking of planar position plus orientation.

3.6.1 Target Localization and Segmentation

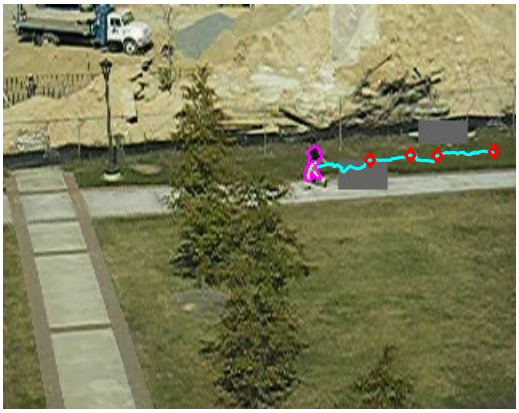
The feature vectors are built using the color and the spatial values of the pixels, ($u = [\mathcal{I}(x), x]$) from three segmented templates of the target object in different postures. The Σ

matrix in the Gaussian kernel (Equation (4)) for learning the target model is diagonal with $\sigma_F = 55$ for the color values and $\sigma_s = 4$ for the spatial domain. The number of eigenvectors used are in the range $m \in [5, 8]$. For computational efficiency, the learned space is reduced using the method presented in Chapter 4. As a result, Equation (17) was used for computing the projections with $l \in [8, 12]$. The proposed tracker was implemented in MATLAB on a Intel Core2 1.86 GHz processor with 2GB RAM. Run-time for the proposed tracker was less than 1 frame/sec for all experiments. Segmentation was achieved by thresholding as per Section 3.5.2.

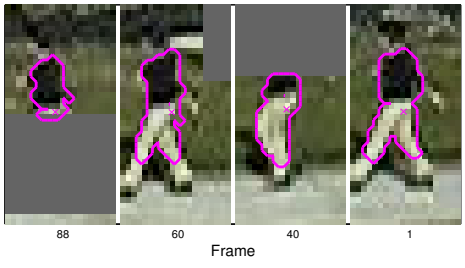
Sequence 1 has artificial occlusions for verification purposes; see Figure 19. First the upper part of the body is occluded than the lower part. The cyan line in Figure 19(a) shows the trajectory of the target and red diamond signs show the location where the snapshots, shown in Figure 19(b), were taken. The projections on the top four eigenvectors are also shown in Figure 19(c). The projections show that different eigenvectors capture different clusters corresponding to different parts of the body. In the first occlusion, the third eigenvector, which captures the upper part of the body, has negligible effect and the projections onto the third eigenvector are close to zero. Similarly in the second occlusion, the second and the fourth eigenvectors, which capture the lower part of the body, have negligible effect on the localization procedure. Eigenvectors corresponding to the visible parts are used to track through the occlusion. Similarly, in Figure 20, we show the projections on top four eigenvectors for sample frames. The proposed tracker successfully tracks through the occlusions.

In Figure 21, the proposed tracker successfully tracks a small target of window size 10×15 in cluttered environment. The video sequence has illumination changes as is evident from the snapshots depicted in Figure 21(b). Throughout the tracking sequence, the segmentation results are accurate and robust to misleading background image content.

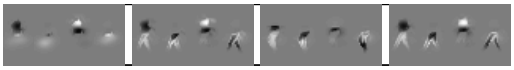
Figure 22 depicts two scenarios with massive occlusions (50% or more) of the track target. Of note, the track pointer location remains consistent during occlusions. Many



(a) Sample frame from Seq 1, resolution 320 × 240. The cyan line shows the trajectory of the target and red diamond signs show the location where the snapshots shown below were taken.

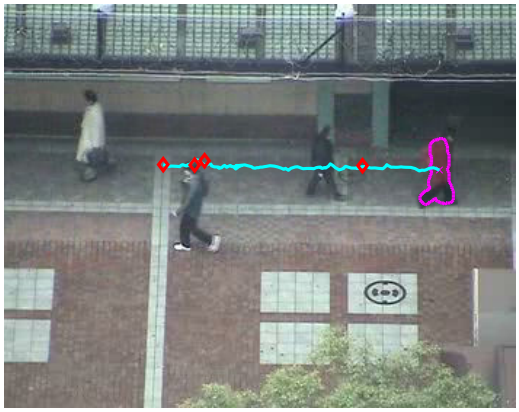


(b) Snapshots, window size 20 × 30

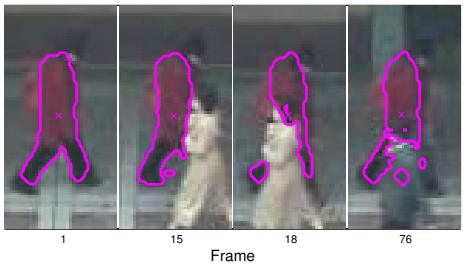


(c) Projection on the first four eigenvectors (from left to right).

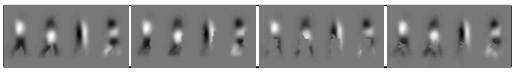
Figure 19: Sequence 1: Video with artificial occlusions. Eigenvectors corresponding to the visible parts are used to track through the occlusion.



(a) Sample frame from Sequence 2, resolution 320 × 240. The cyan line shows the trajectory of the target and red diamond signs show the location where the snapshots shown below were taken.



(b) Snapshots, window size 40 × 80

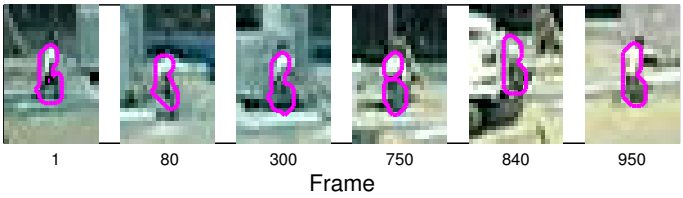


(c) Projection on the first four eigenvectors (from left to right)

Figure 20: Sequence 2. Eigenvectors corresponding to the visible parts are used to track through the occlusion.

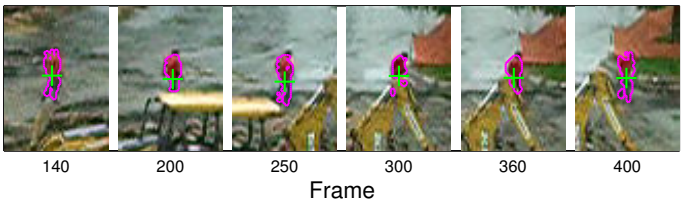
template-based methods and shape-based methods would experience a shift in target location and would not track as smoothly.

(a) Sample frame from Sequence 3, resolution 320 × 240

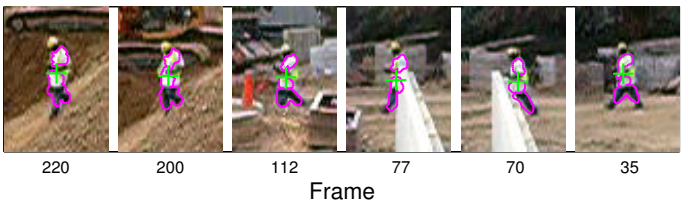


(b) Snapshots, window size 10 × 15

Figure 21: Sequence 3. Tracking and segmenting a small object in cluttered environment with illumination changes.



(a) Sequence 6: resolution 320 × 240, window size 25 × 35.



(b) Sequence 7: resolution 320 × 240, window size 25 × 35.

Figure 22: Sequence 6 and 7. Tracking construction workers in cluttered environment with occlusions.

3.6.2 Target Localization

This section focuses on target localization, for which Figure 23 depicts several test scenarios. Some of the sequences can be downloaded from [1, 2]. Segmentation is not performed and only one template image is used. The feature vectors are constructed using the color and spatial values, i.e., for a pixel at location x the feature vector is $u = [\mathcal{I}(x), x]$. The Gaussian kernel (Equation (4)) matrix Σ is diagonal with $\sigma_F = 60$ for the color values and $\sigma_s = 4$ for the spatial domain. The number of eigenvectors m retained were chosen following [43]. In particular, given that the error associated with the eigenvector k is

$$\epsilon^k = \left\{ \frac{1}{n} \sum_{i=1}^{n} f^k(u_i) \right\}^2 ,$$

the eigenvector indices k satisfying the following inequality were retained,

$$\left\{ \frac{1}{n} \sum_{i=1}^{n} f^k(u_i) \right\}^2 > \frac{1}{1+n} \left\{ \frac{1}{n} \sum_{i=1}^{n} (f^k(u_i))^2 \right\}.$$

In practice, about 25 of the leading eigenvectors were kept, i.e., $m = 25$. Recall that the number of eigenvectors retained while performing localization and segmentation in Section 3.6.1 was $m \in [5, 8]$. Here, a larger number of eigenvectors is kept because the template is a rectangular region that may contain background information. Critical target object information may be thrown out by keeping too few eigenvectors. In Section 3.6.1, manually segmented templates were used to form the input space and therefore contained little, if any, background information.

The proposed method was compared with the ensemble tracker [14], the covariance tracker [73], and the mean-shift tracker [28]. For the ensemble tracker, the 11-dimensional feature vector per pixel consisted of an 8-bin local orientation histogram calculated on 5×5 window as well as the pixel R, G, and B values. Five weak classifiers were trained. For each video sequence, we first ran the tracker by updating the weak classifiers at each frame. During the update, at most two weak classifiers were dropped, followed by the training of two new weak classifiers plus the update of the remaining classifiers. A second instantiation used a static tracker that trained five weak classifiers on the first frame of the sequence, which then remained fixed for the entire length of the sequence. The best result for each

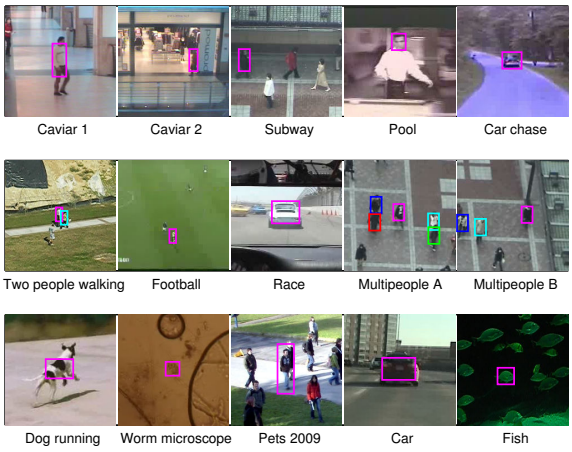


Figure 23: Tracked objects from test sequences.

sequence was kept and reported.

For the covariance tracker [73], the pixel-wise feature vectors were built using the color values, the image gradient, and the spatial location of the pixel, i.e, for a pixel at location x the feature vector is $u = [\mathcal{I}(x), \nabla\mathcal{I}(x), x]$. The target object in the first frame was used as template to build the model covariance matrix. The covariance matrix was updated at each frame according to the method described in [73]. For mean-shift tracker [28], the target in the first frame was used to build the weighted histogram.

The results are shown in Table 2. For the proposed, the ensemble, and the mean-shift trackers, reinitialization was performed if track loss occurred. The table lists the number of reinitialization required to completely track the sequence. For the covariance tracker, the detection rate is listed (the ratio of the number of correct estimates to the total number of frames), because it is a global search method. An $\times$ in the table indicates that the corresponding tracker needed more than 3 reinitializations or had less than 70% detection rate.

As can be seen from the table, the proposed tracker is able to completely track all sequences. For the sequence in which multiple trackers performed well, the matched object region for the proposed tracker is more consistent and accurate than the comparison methods, see Figure 24. Many sequences contain partial occlusions such as the Caviar 2 sequence, whose sample frames are shown in Figure 25(a). The target object is occluded

Table 2: Tracking Results: [1] Number of reinitialization required to complete the tracking. × indicates more than 5. [2] Ratio of correct number of estimation to the total number of frames. × indicates less than 70%.

Sequence	Resolution	O. size	Frames	**Proposed**[1]	Ensemble[1]	MS[1]	COV[2]
Caviar 1	384 × 288	25 × 60	100	**0**	0	1	100
Caviar 2	384 × 288	15 × 40	165	**0**	×	×	×
Subway	352 × 288	20 × 40	171	**0**	×	1	97.4
Face	352 × 240	30 × 40	90	**0**	3	0	91.4
Car chase	640 × 480	35 × 40	800	**0**	0	×	×
Two people	320 × 240	12 × 25	540	**0/0**	1/×	0/0	×/×
Football	320 × 240	12 × 25	225	**0**	0	×	×
Race Car	320 × 240	50 × 40	749	**0**	1	0	99.7
Multi-people A	440 × 360	20 × 30	1195	**0**	0	0	×
Multi-people B	440 × 360	20 × 30	480	**0**	0	0	×
Dog Running	352 × 240	50 × 35	123	**0**	1	0	92.8
Worm	320 × 240	25 × 25	300	**0**	×	0	×
Pets 2009	320 × 240	25 × 25	300	**0**	×	0	×
Car	640 × 480	60 × 40	286	**0**	0	0	100
Fish	320 × 240	30 × 30	205	**0**	0	2	×

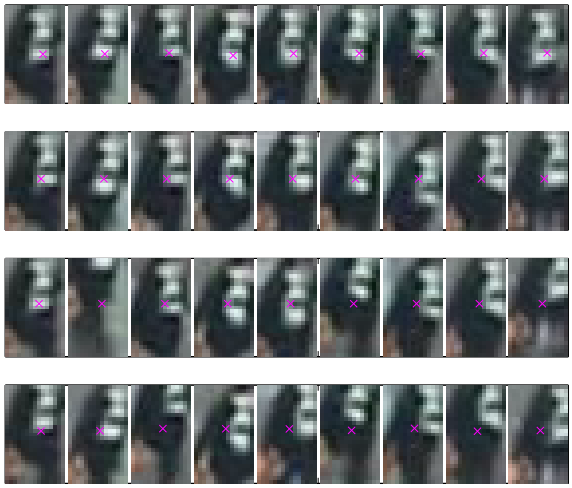
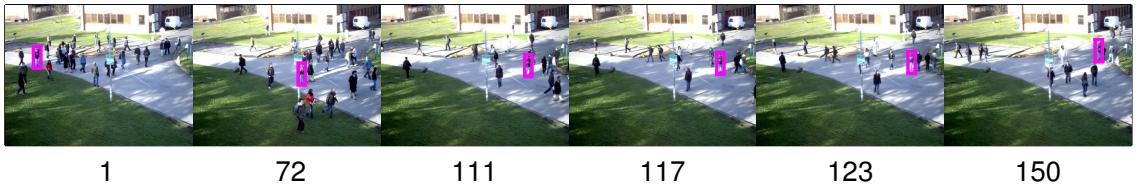


Figure 24: Matched object regions for a target in the sequence Multipeople A using the proposed, ensemble, covariance and mean-shift trackers (top to bottom). The track point is more stable in the proposed methods.

(a) Caviar data set.



(b) PETS 2009 data set.

Figure 25: In the Caviar data set, the target object is occluded by people coming from the opposite direction. In PETS 2009 data set, the target object is successfully tracked in a crowded scene.

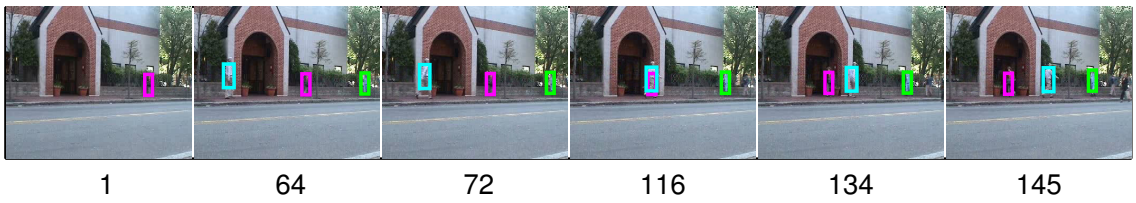
by people coming from the opposite direction. Similarly, the target object in PETS 2009 data set (Figure 25(b)) is successfully tracked in a crowded scene until the person is fully occluded.

3.6.3 Target Localization on IR Sequences

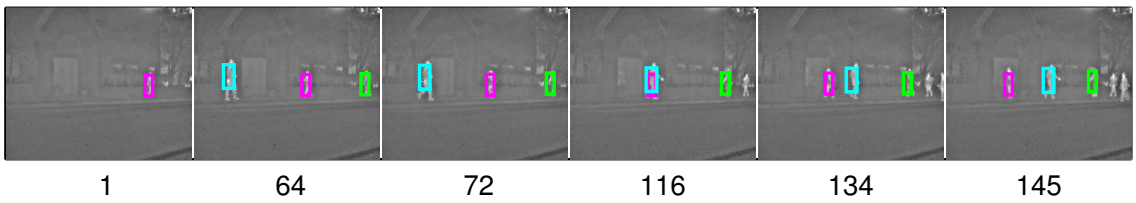
The proposed algorithm was also tested on a combined visible-infrared sequence. In Figure 26, the feature vectors per pixel are formed using the color, infrared and spatial values, i.e., $u = [\mathcal{I}(x), IR(x), x]$. Tracking on only color or only infrared information resulted in loss of track, but by combing both the color and infrared, the entire sequence could be processed. For this sequence, the covariance tracker had the detection rate of 96% using both the color and infrared sequences [73].

3.6.4 Orientation Tracking

The proposed tracker was also used to track the orientation. For orientation tracking, the gradient update Equation (20) was used. Some algorithms presume independence of the spatial and appearance models for simplicity, thus losing some or all ability to track orientation. The proposed algorithm faithfully encodes for any correlations between the spatial and appearance information. Figure 27 shows that the proposed method can track the



(a) Color sequences



(b) Infrared sequences

Figure 26: Tracking of visible-infrared sequence. The feature vectors per pixel are formed using the color, infrared, and spatial values, i.e., $u = [\mathcal{I}(x), IR(x), x]$.

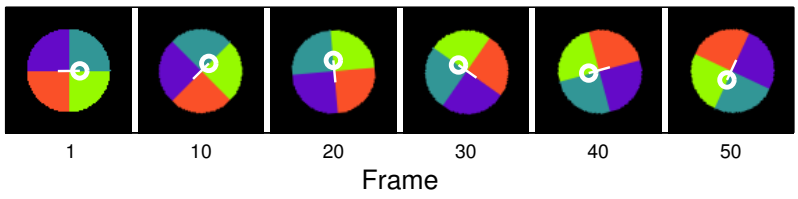
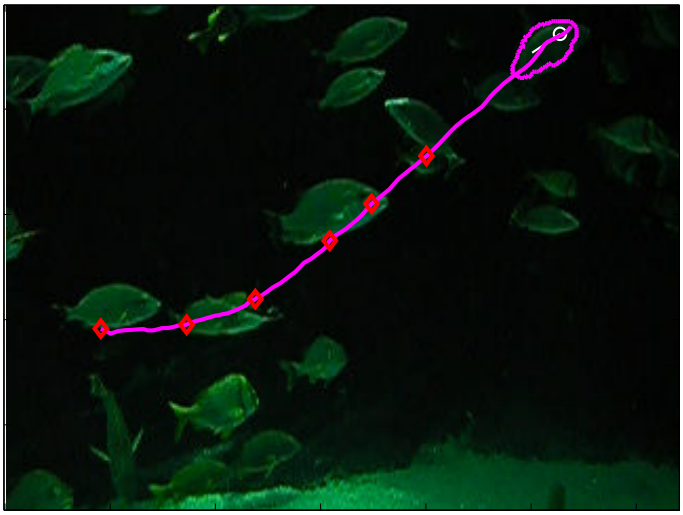


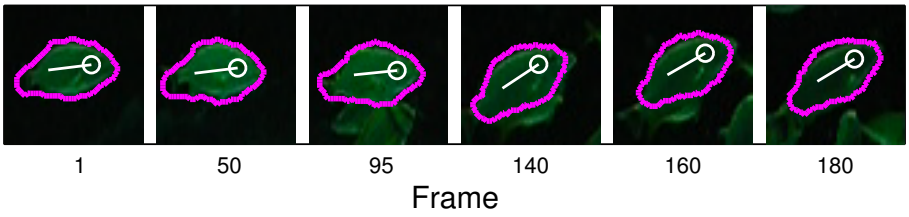
Figure 27: Beach ball orientation tracking. White line with circle indicates the orientation.

orientation of a beach ball, whose shape is rotationally invariant, but whose appearance is not. Such tracking is not possible when there is independence of the feature and appearance information.

The tracker is applied to two other sequences with the results depicted in Figures 28 and Figure 30. Note that the fish sequence has both distracting false positives in the form of other fish and illumination change. The two main challenges in the worm sequence are illumination changes and rapid orientation changes. During tracking the worm performs several complete rotations. For both sequences ground truth comparison of the position and orientation is shown in Figures 29 and 31.



(a) Sample frame from fish sequence (Image size 320×240). Cyan line indicates the path traversed and red diamonds indicate places where snapshots (shown below) were taken.



(b) Snapshots. White line with circle indicates orientation.

Figure 28: Tracking of a fish sequence.

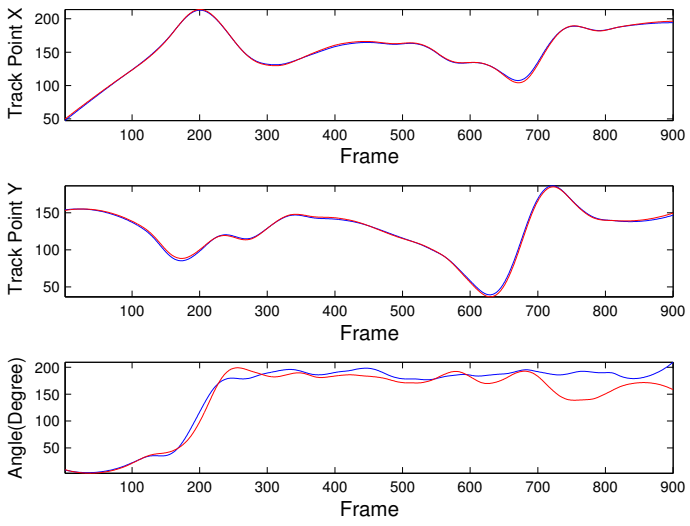
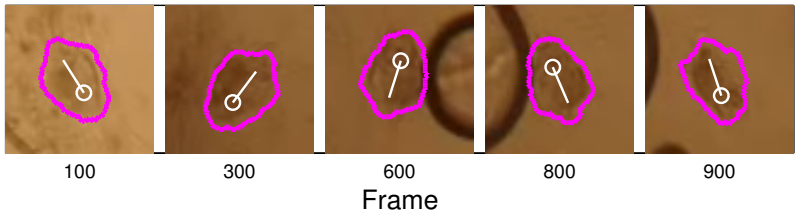


Figure 29: Ground truth comparison for fish sequence. Red: Ground truth, Blue: Proposed tracker.

(a) Sample frame (Image size 320×240).



(b) Snapshots. White line with circle indicates the orientation.

Figure 30: Tracking of a worm under microscope.

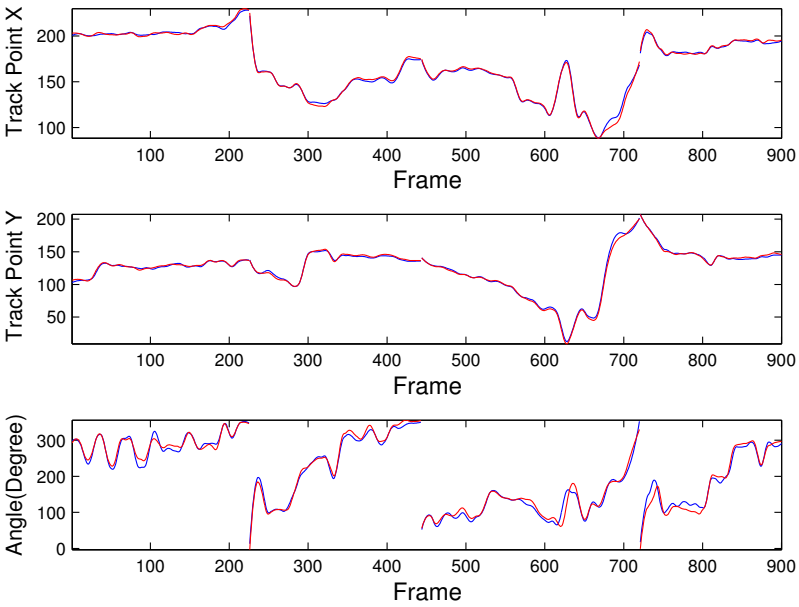


Figure 31: Ground truth comparison for worm sequence. Red: Ground truth, Blue: Proposed tracker.

3.7 Conclusion

This chapter presented a KPCA-based eigenspace object tracking algorithm. KPCA was used for the eigenspace representation due to its nonlinear characteristic, excellent discrimination capability, noise suppression, and outlier rejection. A similarity function was derived to measure the similarity of pixel vectors to the learned space. The similarity function has advantageous statistical and robustness properties. Target localization was carried out as a global detection process for a 1D synthetic example, and using a variational approach for image sequences. Derivations are given for both gradient- and mean shift-based optimization procedures to maximize the similarity measure in the KPCA eigenspace. The battery of test sequences and comparison with existing methods shows improved tracking performance.

CHAPTER IV

KERNEL MAP COMPRESSION

4.1 Introduction

This chapter presents a method for improving the computational and representational efficiency of algorithms based on Mercer kernel methods. Mercer kernel methods generate powerful techniques for studying non-linear data [87]. Many useful linear algorithms involving dot products can be made nonlinear by employing Mercer kernels (see Chapter 2). The method proposed in this chapter can be used in all kernel-based methods, and the dimensional reduction and manifold reduction methods mentioned in Chapter 2. However, this chapter focuses on two of such algorithms: kernel principal component analysis (KPCA) [86] and support vector machines (SVM) [90]. The former is an unsupervised learning algorithm, while the latter is a supervised learning algorithm. They have been applied in numerous image processing and computer vision applications, see for example [13, 66, 87, 112]. The superior performance of Mercer kernel-based algorithms comes at a price of increased storage and computational requirements [84].

Accurately learning the desired functional relationship in the training phase requires a large number of training samples. The large training set presents a difficulty, since it requires that the whole input space associated to the training set be stored and processed at once. Methods have been proposed for both the KPCA and the SV machines to learn the space incrementally [27, 46, 58, 106]. However, once the space has been learned, either incrementally or in batch, all subsequent computations in the high-dimensional space are performed through the kernel in terms of linear combinations of all training vectors. For massive datasets this still requires high storage requirements and computational complexity, making kernel methods unfavorable for on line computer vision applications.

Schölkopf et al [85] discuss two classes of solutions to reduce the execution space. The first class, called *reduce set selection* (RSS), finds a reduced set of expansion vectors from

the original space that approximates well the training set. RSS has been shown to work well (see refs. in [61, 85]. A related selection method, kernel matching pursuit, is discussed in [103]. The second class, called *reduce set construction* (RSC), identifies new elements of the input space that approximate well the training set. RSC has improved compression versus RSS [85], but has a more expensive upfront cost to find the reduced set. Related methods applicable only to SVMs include [53], which is a basis selection method, and [17], which extends [85] to incorporate shared support vectors. Other methods that reduce the space include [97], where a maximum-likelihood approach is used to obtain a sparse representation. However, the method is not general and is used for KPCA only.

A related form of learning is neural networks [39, 72]. Neural networks have the property of universal approximation [71, 81]. Further, they are more efficient than kernel methods [85]. Thus, in principal, neural networks should be capable of providing efficient representations of the functions or mappings arising from Mercer kernel methods, however previous attempts have not been so fruitful [88].

4.2 Contribution

This chapter exploits the relation between kernel methods, regularization theory and radial basis functions to propose a novel method to reduce the computational complexity associated with kernel methods. The work differs from previous efforts in that we approximate the learned function in a manner more compatible with the universal approximation capabilities of neural networks. In particular, once the function or mapping is learned using kernel methods, we identify the underlying algorithmic step involving the empirical kernel map most suited to neural network approximation, which is the input to KPCA-space mapping. The final algorithm is a two-step process beginning with the learning procedure, followed by the compression procedure. Efficient representations are achieved with minimal loss of performance.

4.3 Existing Efforts

Let $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ be the input space associated to a sample set.

Kernel principal component analysis Following the procedure in Chapter 2, given a

Mercer kernel $\boldsymbol{k}$, the KPCA eigenvectors are given by $V^k = \sum_{i=1}^{n} \alpha_i^k \phi(x_i)$, where $\alpha_i^k = \frac{a_i^k}{\sqrt{\lambda^k}}$, and $a_i^k, i = 1, \ldots, n$ and λ^k are the k-th eigenvector and eigenvalue of the Gram matrix K. A test point x is represented in the KPCA space by projecting it onto the eigenvectors V^k. The projection on the k-th eigenvector is

$$f^k(x) = V^k \cdot \phi(x) = \sum_{i=1}^{n} \alpha_i^k \boldsymbol{k}(x_i, x). \tag{22}$$

Support Vector Machines (SVM) is a supervised learning algorithm. Given a set of input/output sample pairs, $\{(x_i, y_i) | x_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}_{i=1}^n$, associated to a function, an SVM approximates the function using the following form:

$$f(x) = V \cdot \phi(x) = \sum_{i=1}^{n} \alpha_i \boldsymbol{k}(x_i, x) + b, \tag{23}$$

where $V = \sum_{i=1}^{n} \alpha_i \phi(x_i)$. The contributions of $\phi(x_i)$ with $\alpha_i \neq 0$ are called *support vectors*.

SVM is also used to perform classification by constructing a high-dimensional hyperplane that separates the data into two categories. The nonlinear decision boundary is given by $\text{sgn}(f(x))$. For each class to identify, there is an associated support vector, thus leading to a collection of support vectors $\left\{V^1, ..., V^{n_c}\right\}$ whose coefficient vectors are $\left\{\{\alpha_i^1\}_{i=1}^n, \ldots, \{\alpha_i^{n_c}\}_{i=1}^n\right\}$, where n_c is the number of classes.

Reduced Set Methods To carry out either the projections (Equation (22)) or classification (Equation (23)), the computation depends on the entire input space $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ (or support space for SVM), leading to the high storage and computational requirements. Schölkopf et al [85] propose two solutions to reduce the space. The first, called *reduce set selection* (RSS), deals with the problem of how to select a reduced set of expansion vectors from the original space so that the Equations (22) and (23) are represented in terms of the reduced expansion vectors. The second, called *reduce set construction* (RSC), constructs new vectors to achieve high reduction rates. RSC gives better compression results than RSS [85], but with a higher computation cost.

Both reduced set methods, RSC and RSS, obtain a reduced kernel mapping through the approximation of the vectors V^k by

$$V^{'k} = \sum_{j=1}^{l} \beta_j \phi(z_j), \tag{24}$$

where $l < n$ (for the SV case $l < n_c$), $z_j \in \mathbb{R}^d$ are the new input vectors, and the $\beta_j \in \mathbb{R}$ are their coefficients [84]. The coefficients β_j and new input vectors z_j are found by minimizing

$$\left|\left|V^k - V^{'k}\right|\right|^2. \tag{25}$$

The reduction procedure starts with $l = 1$ and, instead of minimizing Equation (25), maximizes the following equivalent function:

$$\frac{(\tilde{V}_j \cdot \phi(z_j))^2}{(\phi(z_j) \cdot \phi(z_j))},$$

where $\tilde{V}_1 = V^k$, and the $\tilde{V}_j$ for $1 < j \leq l$ is defined below.

To find the pre-image, z_1, fixed point or gradient based optimization procedures are used [84, 24]. Once the optimal pre-image z_1 is found, the coefficient β_1 is determined by setting $\beta_1 = \frac{(\tilde{V}_1 \cdot \phi(z_1))}{(\phi(z_1) \cdot \phi(z_1))}$. To find the next pre-image vector z_{j+1}, define $\tilde{V}_{j+1} := \tilde{V}_j - \beta_j \phi(z_j)$, where $\tilde{V}_j = \sum_{i=1}^{n} \alpha_i \phi(x_i) - \sum_{b=1}^{j-1} \beta_b \phi(z_b)$, and z_1 by z_{j+1}. In [24], once all the z_j and β_j are found, the procedure is followed by simultaneous optimization over all z_j and β_j. The procedure is performed to approximate all V^k, each with its own set of pre-image vectors $\{z_j^k\}$ and coefficients $\{\beta_j^k\}$, where the superscript of k specifies the connection to V^k.

4.4 Kernel Map Compression

The previous section described the reduced set method as an iterative procedure that identifies the input space pre-image vectors $\{z_j^k\}$ and associated coefficients $\{\beta_j^k\}$, $j = 1 \ldots l$ that approximate well the original Hilbert space vector(s) V^k. We propose a novel method to cull the input sample space by exploiting the relationship between (neural network) regularization theory and kernel methods [39, 93]. Previous methods try to approximate the vectors V^k in the Hilbert space (see Equation (24)), so that the projections f (blue arrow in the Figure 32) are computed using the approximated vectors. We, on the other hand, try to approximate the empirical kernel maps $V^k : \mathbb{R}^d \rightarrow \mathbb{R}$ directly, circumventing the Hilbert

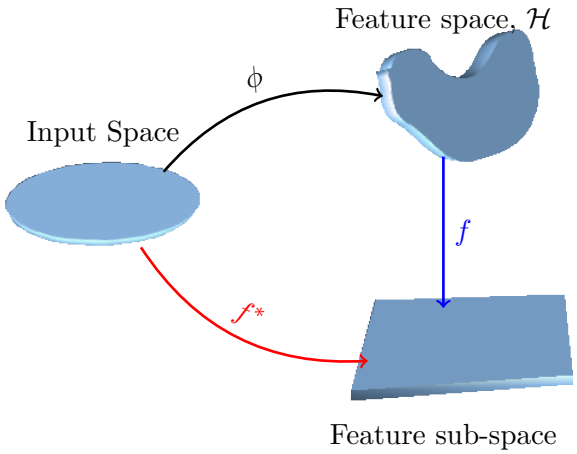


Figure 32: Kernel map compression (KMC). The red arrow shows the proposed approach to approximate the relationship between the input space and the feature sub-space directly, circumventing the feature space.

space. The proposed approach is explained in the Figure 32, where the red arrow depicts our approach to approximate the relationship between the input and the feature sub-space using generalized radial basis functions (GRBFs).

In the finite-dimensional feature sub-space of $\mathcal{H}$, we have that the k-th coordinate of the image of x_i is

$$y_i^k = V^k(x_i) = \sum_{i=1}^{n} \alpha_i^k \boldsymbol{k}(x_i, x). \tag{26}$$

This input-output equation is found in both KPCA (22) and SVM (23), and is determined during the KPCA and SVM learning procedure. Further, the input-output relationship between x_i and y_i is much simpler than that of x_i and V^k. In what follows, we describe how the collection of all $\{x_1, \ldots, x_n\}$ and associated $\{y_1^k, \ldots, y_n^k\}$ are learned for each k. In particular, we point out how the function in Equation (26) is of the class of functions that are efficiently approximated by generalized radial basis functions [81]. *Without loss of generality, and for purposes of exposition, we will ignore the superscript* $\cdot^k$ *in what follows, then introduce it again later.*

4.4.1 Setup

In regularization theory [72], the approximation problem is described as follows: *Given* n *different points* $\{x_i \in \mathbb{R}^d, i = 1, \ldots, n\}$ *and* n *real numbers,* $\{y_i \in \mathbb{R}, i = 1, \ldots, n\}$*, find a*

function $f : \mathbb{R}^d \to \mathbb{R}$ *that minimizes*

$$H[f] = \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \, ||Pf||^2 , \tag{27}$$

where P *is a constraint operator (usually a differential operator),* $||\cdot||$ *is a norm on the function space, and* λ *is the regularization parameter.* If $||Pf||^2$ is rotationally and translationally invariant, the regularized solution is given by the expansion in radial basis functions,

$$f(x) = \sum_{i=1}^{n} w_i r(|x_i - x|).$$

where $|\cdot|$ is a norm on the input space and r is the radial basis function (RBF). The RBF r is a real valued function depending only on the distance of the input from the origin. Commonly used types of RBFs include Gaussian, multiquadric, thin plate spline and polyharmonic spline. The function f is given by the sum of n RBFs, placed at points $\{x_i\}_{i=1}^n$ and weighted by coefficients $\{w_i\}_{i=1}^n$. The coefficients are found using the interpolation conditions $f(x_i) = y_i, i = 1, \ldots, n$. In many practical applications P is rotationally and translationally invariant.

In the RBF approach, the function f is expanded on a set of radial functions r centered at data points. The function is then a point in a multidimensional space, whose dimension is equal to the number of data points. This indicates that the function f can be approximated by an expansion in a basis with a smaller number of dimensions [72],

$$f^*(x) = \sum_{j=1}^{l} \gamma_j r(|c_j - x|), \tag{28}$$

where c_j and γ_j are $l < n$ center points and their coefficients. Now the problem consists of finding the optimal c_j and γ_j. RBFs with movable centers are called generalized radial basis functions (GRBFs).

4.4.2 Procedure

In what follows, we describe the procedure for generating the GRBF approximation to the input-output set $\{(x_1, y_1) \ldots, (x_n, y_n)\}$.

Finding optimal coefficients γ_j: Starting from an initial set of center locations c_j, the coefficients γ_j, can be found by imposing the constraints $f^*(x_i) = y_i$, leading to the

following system of over constrained linear equations:

$$y_i = \sum_{j=1}^{l} \gamma_j r(|c_j - x_i|) \ , i = 1, \ldots, n.$$

The least squares solution determines an approximate solution to the overdetermined system. The least squares formula is written as

$$\boldsymbol{\gamma} = (R^T R)^{-1} R^T \boldsymbol{y}, \tag{29}$$

where $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_l]^T$ is an l-dimensional coefficient vector, R is the $n \times l$ matrix with entries $R_{ij} = r(|x_i - c_j|)$, and $\boldsymbol{y} = [y_1, \ldots, y_n]^T$. The $l \times l$ matrix $R^T R$ is nonsingular [72] and therefore invertible.

Finding optimal center location c_j: To find the optimal center locations c_j, minimize the Equation (27), with f replaced by f^*. Different optimization algorithms can be used for this purpose. We show here calculations using gradient descent and setting $\lambda = 0$. We start with the initial estimation of l center points and use the following equation to find the update:

$$c_j(t+1) = c_j(t) + \delta t \frac{\partial H[f^*]}{\partial c_j} \ j = 1, \ldots, l, \tag{30}$$

where δt is the time step and $\frac{\partial H[f^*]}{\partial c_j}$ is given by

$$\frac{\partial H[f^*]}{\partial c_j} = -4\gamma_j \sum_{i=1}^{n} (y_i - \hat{y}_i) r(|c_j - x_i|)(c_j - x_i),$$

where $\hat{y}_i = f^*(x_i)$.

Since the method described above is based on gradient descent, the starting locations of c_j need to be estimated. We can start from equally spaced center locations spread over the original space, or use the center locations found as a result of RSS [85] as the starting center locations. In the next subsection we give two algorithms that differ in how we place the initial centers.

4.4.3 Choosing Initial center locations.

Algorithm 1: In algorithm 1, we either uniformly chose l centers or use k-means clustering to find l centers as starting locations. In summary, the following steps are taken to find the reduced space:

1: Select l initial center positions c_j for the GRBFs.

2: Use Equation (29) to find the coefficients γ_j.

3: Iterate Eqs. (29) and (30) until the error given by Eq. (27) goes below some threshold value.

Algorithm 2: Algorithm 1 can be applied to each of the functions V^k and their associated input-output sets $\{x_1, \ldots, x_n\}$ and $\{y_1^k, \ldots, y_n^k\}$. However when V^k corresponds to eigenvectors of KPCA, the approach of using k-means to find the initial center positions may not be suitable. Because different eigenvectors capture different clusters of the input space, placing the initial centers using k-means clustering may lead the algorithm to get stuck in a local minimum. To avoid this issue we use the following steps.

1: Start with one GRBF center given by

$$c_1 = p = \underset{p \in \{p_i\}_{i=1}^n}{\arg\max} \left| V \cdot \phi(p) \right|,$$

i.e., choose a point from the input space that maximizes the projection onto the target eigenvector. Equation (29) is used to find the coefficients γ_1.

2: Add $l - 1$ new GRBFs at

$$c_i = p = \underset{p \in \{p_i\}_{i=1}^N}{\arg\max} \left(y_i - f^*(p) \right)^2,$$

where $f^*(p)$ is given by Equation (28).

3: Iterate Eqs. (29) and (30) until the error given by Eq. (27) goes below some threshold value.

In this algorithm, once the initial center locations for the GRBFs are chosen, all center locations and coefficients are optimized simultaneously. The techniques contrasts with Schölkopf [84], where optimization is performed one point at a time. In Burges method [24], optimization is performed one point at a time, followed by a second phase with all parameters optimized jointly.

4.4.4 Approximating Several Functions at Once

Typical applications require approximating more than one function in the feature space. For example, we would like to efficiently compute all the functions $V^k(x)$, corresponding to

$f^k(x)$ in Equation (22), and the collection of binary classifiers $V^k(x)$ arising from Equation (23). Approximating each function separately using the method described in Section 4.4 may not result in adequate compression. The procedure will return a pair of center locations and their coefficients for each function V^k, $\{c_j^k, \gamma_j^k\}_{j=1}^l$, where $k = 1, \ldots, m$ and m is the total number of functions to be approximated (varies based on KPCA vs SVM). A more efficient approach is to share center locations c_j for all the functions V^k with different coefficients γ_j^k. The functions V^k are then approximated by

$$V^k(x) = \sum_{j=1}^{l} \gamma_j^k r(|c_j - x|).$$

The combined optimization gives a different update method for the coefficients and center locations. To find the coefficients γ_j^k, Equation (29) is used except that the $\boldsymbol{\gamma}$ and $\boldsymbol{y}$ are $l \times m$ matrices, where l is the total number of centers and m is the number of functions to be approximated.

To find the optimal center locations c_j, the functional to be minimized is of the form

$$H[f^{k*}] = \sum_{k=1}^{m} \sum_{i=1}^{n} \left(y_i - f^{k*}(x_i) \right)^2 .$$

4.4.5 Achieving Further Compression

The method achieves compression by only optimizing over the location and the weights of the basis functions, with all other parameters of the basis functions equal. For Gaussian RBFs, additional modifiable parameters include the positive symmetric operator defining the norm $|\cdot|$ for each RBF, which affects the orientation and anisotropy associated to the RBF. Optimizing over these parameters may lead to further reductions in the number of RBFs needed [72].

4.4.6 Computational Cost

The proposed method iterates between finding the updated coefficients γ_i and the centers c_i, until the error given by Equation (27) goes below some threshold. The equations to iterate are Equations (29) and (30). Equation (29) requires taking the inverse of one $l \times l$ matrix, with the computational cost of $\mathcal{O}(l^3)$, where l is the number of reduced center points c_i.

The computational cost updating the center positions for each c_i (Equation 30) is $\mathcal{O}(l \times n)$. This is the same cost as the RSC methods, since Equation (4) and the discussion following Equation (6) indicate that the principal computations of the RSC method are also of the order of $\mathcal{O}(l \times n)$ and $\mathcal{O}(l^3)$ for optimizing the center locations and coefficients. However when RSC method is followed by phase 2, in which the optimization is performed over all parameters, the computational complexity of RSC method increases by about two orders of magnitude [85].

4.4.7 Pre-image Computations

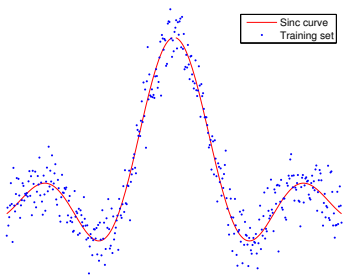
The ideas developed in this chapter can be directly applied to the out-of-sample extension and the pre-image methods of Chapter 2. This will result in computation savings in computing the out-of-sample extensions and the pre-images. The embedding/projection Equation (Equation (5)) can be approximated by the generalized radial basis functions approach of this chapter resulting in the Equation (28).

4.5 Application

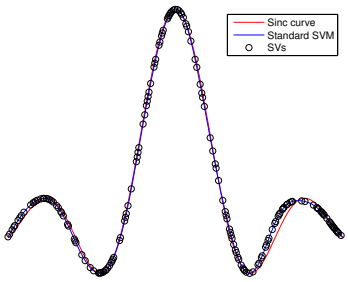
This section describes several experiments validating the proposed kernel compression method. The results are compared to the use of the full training set for evaluation, as well as to comparable algorithms [24, 84]. The first set of experiments utilize low-dimensional, synthetic datasets for visualization purposes, while later experiments utilize more complex, publicly available datasets. The experiments cover application of the proposed method to both SVM and KPCA.

4.5.1 Synthetic Datasets

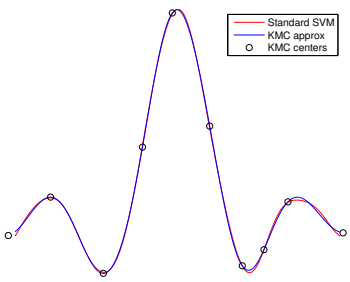
Validation for SVM: To gauge SVM compression, consider the approximation of the one-dimensional function $y = \text{sinc}(x)/x$. A total of 1200 points are uniformly sampled over the domain $[-10, 10]$, 400 of which are taken as the training examples and the others as testing examples. The training samples are corrupted with Gaussian noise $\mathcal{N}(0, 0.1)$. The kernel σ, both for SVM learning and KMC is 1.5. Figure 33(a) shows the corrupted training data along with the true curve. SVM produced 170 support vectors which are shown in Figure



(a) Synthetic dataset for support vector regression. In this experiment the blue data points are used as training data to learn the *sinc* curve shown in red.



(b) ∘ represents the support vectors (SVs). Blue curve is the *sinc* function learned by the standard SVM.



(c) ∘ represent the GRBFs used to approximate the SVM. Blue curve is the *sinc* function learned by the proposed KMC procedure.

Figure 33: Sinc curve synthetic experiment for SVM regression.

33(b). The approximated curve using standard SVM is also shown. The support vectors are reduced using the kernel map compression (KMC) (Figure 33(c)), and two reduced set construction (RSC) methods, FP [84] and Burges [24]. The RSC methods are performed using the Matlab code publicly available for download [3]. The proposed method (KMC) was also implemented using Matlab.

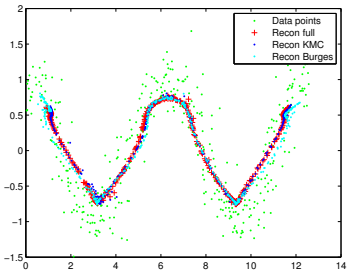
The results are tabulated in the Table 3. The first column of the table lists the algorithms (method - p, where p stands for number of support vectors or number of reduced data points). The second column list the time taken to reduce the support vectors for each algorithm. The third column shows the compression ratio, which is the ratio of full space to the reduced space. The second to last column shows the error in approximating the true sinc curve. The last column shows the percentage degradation of the approximation methods with respect to the standard SVM. As can be seen in the Table 3, the proposed method is able to reach the performance of standard SVM by using just 7 data points (compression ratio of 24) with degradation of less than 5%. For the same compression ratio, the other methods perform worse with degradation of more than 250%. The proposed method also demonstrates improved compression times. Lastly, for modest compression ratios, the KMC methods outperforms SVM.

Validation for KPCA: For KPCA, we apply the proposed method to a simple 2D example. The first six eigenvectors obtained as a result of applying KPCA to a synthetic

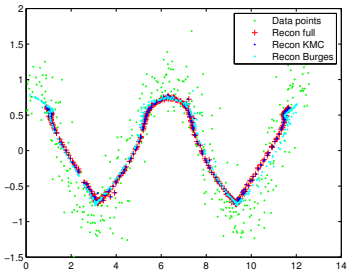
Table 3: Performance comparison of approximation methods for approximating the sinc curve. method - p means that the corresponding method used p support vectors or reduced data points to approximate the sinc curve. The proposed method (KMC) is able to reach the performance of standard SVM by using just 7 data points (compression ratio of 24) with degradation of less than 5 %

Algorithm	Compression time (sec)	Compression ratio	Error	Degradation
SVM - 170	NA	1	.00041	-
KMC - 5	1.82	34	.00098	139%
KMC - 6	2.18	28.3	.00073	78%
KMC - 7	**2.36**	**24.2**	**.00043**	**4.8 %**
KMC - 10	4.38	17	.00037	-9.7%
KMC - 12	4.44	14.2	.00036	-12.2%
FP - 5	4.95	34	.0018	339%
FP - 6	86.85	28.3	.0014	241%
FP - 7	30.58	24.2	.0015	265%
FP - 10	11.83	17	.00046	12.2%
FP - 12	75.26	14.2	.00042	2.4%
Burges - 5	16.24	34	.003	631%
Burges - 6	17.65	28.3	.0019	363%
Burges - 7	22.19	24.2	.0018	339%
Burges - 10	30.79	17	.00089	117%
Burges - 12	35.48	14.2	.00068	65.8%

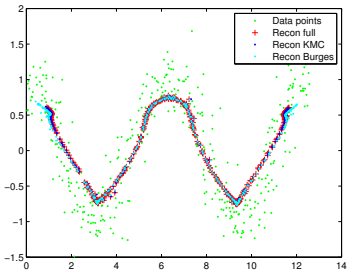
dataset of 400 data points (shown in red in Figure 34(a)) are approximated using $p \in \{6, 8, 12, 15, 20\}$ points. For KPCA, the Gaussian kernel is used with $\sigma = 1$. The same kernel is used for the KMC method. The average error in projecting 400 data points onto each of the the six eigenvectors using p points is shown in 35(a). The error in case of the KMC method is lower than the Burges method. We also compute the average reconstruction error of the 400 data points using six eigenvectors, each one approximated using p points. The reconstruction algorithm of [75] was used. Figure 35(b) shows the error in reconstructing the data set using the reduced space for both methods. The error remains below the Burges method at the compression rates tested. The reconstructed data points for the case of $p \in \{6, 8, 12\}$ are shown in Figure 34, where the original 400 data points are shown in green, the reconstructed data points with six eigenvectors using full space and using only 6 points are shown in black, blue (KMC) and cyan (Burges [24]). The KMC reconstruction using only 6 data points more closely follows the reconstruction using the



(a) Data set and reconstructed data set using 6 eigenvectors, each one approximated using 6 points.

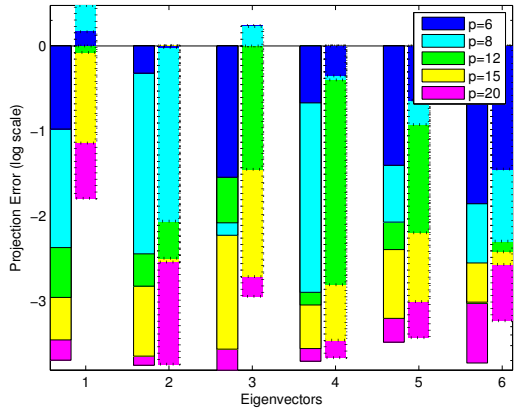


(b) Data set and reconstructed data set using 6 eigenvectors, each one approximated using 8 points.

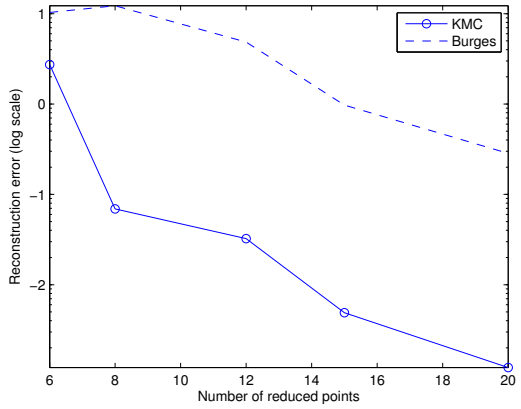


(c) Data set and reconstructed data set using 6 eigenvectors, each one approximated using 12 points.

Figure 34: Performance comparison for the synthetic 2D example for KPCA.



(a) Average error in projecting 400 points onto six eigenvectors, each one approximated using p points. Solid line: KMC, dashed line: Burges



(b) Reconstruction error vs the number of points the space is reduced. Solid line: KMC, dashed line: Burges

Figure 35: Performance comparison for the synthetic 2D example for KPCA.

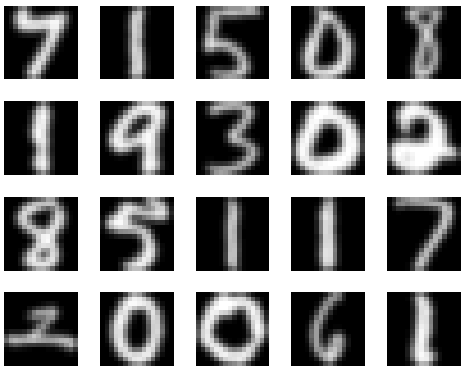
full input space. The results using FP method [84] are not shown as they produced worse results than the Burges method for this example.

4.5.2 Speeding up Support Vector Machines

We applied the proposed method to two real-world classification databases. One is a character recognition data set, downloadable from the UCI machine learning repository [12]. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 images. Each image was then converted into a 16D vector (see Figure 36(a)). The database was divided into 16,000 items for training with the remaining 4,000 cases used for testing. A total of 26 binary classifiers were trained

	Attribute Information		Characters		
			T	D	N
1.	x-box	horizontal position of box	2	4	7
2.	y-box	vertical position of box	8	11	11
3.	width	width of box	3	6	6
4.	high	height of box	5	8	6
5.	onpix	total # on pixels	1	6	3
6.	x-bar	mean x of on pixels in box	8	10	5
7.	y-bar	mean y of on pixels in box	13	6	9
8.	x2bar	mean x variance	0	2	4
9.	y2bar	mean y variance	6	6	6
10.	xybar	mean x y correlation	6	10	4
11.	x2ybr	mean of x * x * y	10	3	4
12.	xy2br	mean of x * y * y	8	7	10
13.	x-ege	mean edge count left to right	0	3	6
14.	xegvy	correlation of x-ege with y	8	7	10
15.	y-ege	mean edge count bottom to top	0	3	2
16.	yegvx	correlation of y-ege with x	8	9	8

(a) Sample 16D descriptor vectors from the character recognition database. The data set consists of 20,000 instances.



(b) Samples from USPS handwritten digits database, composed of 9298 images of dimensions 16×16.

Figure 36: Samples for testing reduced SVM.

to distinguish each character from the rest.

The second dataset is the USPS database of handwritten digits [85], which consists of 9298 handwritten digits of dimensions 16×16. The dataset was randomly shuffled and divided into training and test set consisting of 4649 cases each. A total of 10 binary classifiers were trained, one for each digit. Samples from both the databases are shown in Figure 36. For classification, the Gaussian kernel, with $\sigma = 16$ and $\sigma = 10$, was used (on both databases). The classifiers were approximated using the proposed method (KMC) and the RSC methods of Burges [24] and FP [84]. The Burges method was followed by a second phase, which performed global gradient descent in the space of all parameters (z_i and β_i).

Character recognition: Results for character recognition data set are tabulated in Table 4 for sample letters. The first row lists the characters being classified. The second row shows the original number of support vectors, while the third row shows the number of misclassified characters by the corresponding classifier using the full space. The fourth row onwards lists the number of misclassifications by the proposed and the comparison compression methods. Method-p% means that for each classifier, the space was reduced to p% of the original support vectors. The last two columns list the total % misclassifications and the % degradation in performance as compared to the SVM. The error rates are lower than the competing RSC methods. Performance degradation versus compression ratio is

Table 4: Character recognition database. Top: number of SVs for the original SVM and the number of test errors for each classifier. Bottom: number of test errors for each reduced classifier. Method - p% means that for each classifier, the space was reduced to p% of the original space. Second to last column shows error rate across all 26 classifiers. The last column shows the % degradation.

Characters	A	C	E	H	K	N	P	W	Z	Error	Dgrd
#SV's	154	233	314	484	341	253	239	181	156	-	-
SV	5	12	20	45	24	17	20	6	11	0.39%	0
KMC-3%	36	71	87	110	86	97	64	43	64	1.81%	366%
KMC-6%	24	52	50	74	54	56	54	17	41	1.22%	215%
KMC-9%	12	44	38	105	53	39	42	18	18	0.98%	153%
KMC-15%	13	22	48	85	37	42	40	10	14	0.77%	99%
KMC-20%	6	13	21	51	26	19	21	17	13	0.57%	48%
BG-3%	111	129	164	156	132	170	122	117	105	3.96%	919%
BG-6%	62	131	104	332	87	107	383	154	64	3.00%	674%
BG-9%	34	154	44	160	68	54	77	28	22	1.86%	378%
BG-15%	17	26	71	111	42	109	65	9	20	1.61%	315%
BG-20%	10	20	61	87	26	67	39	27	21	1.07%	176%
FP-3%	154	536	391	1369	501	437	784	1773	172	13.15%	3287%
FP-6%	65	369	199	451	170	230	324	1393	199	7.99%	1957%
FP-9%	249	806	236	364	465	344	163	847	74	7.96%	1951%
FP-15%	91	172	179	355	143	338	161	65	177	4.20%	981%
FP-20%	203	90	232	225	48	55	196	116	175	3.53%	810%

plotted in Figure 37; the degradation curve for the proposed method has a lower slope than the other methods. The bar plot in Figure 37 shows the number of misclassifications when the space is reduced to 20% of the original space for each letter (Last three rows of the Table 4). Burges was followed by a second phase, which increased the compression performance but at increased computational cost. The runtime for compressing the 26 classifiers for the case of 20% reduced space is shown in Figure 38(b). FP has lower runtime but its performance was not good at the compression rates tested. Burges computational cost increases because of the second phase. KMC performed the compression of the complete set about two times faster than Burges method and produced better results.

USPS hand written digits recognition: The results for the USPS hand written database are shown in Table 5. The first row shows the letter being classified. The original number of support vectors are shown in the second row of the table. The third row indicates the number of misclassified digits by the 10 classifiers. The "method - p" indicator means

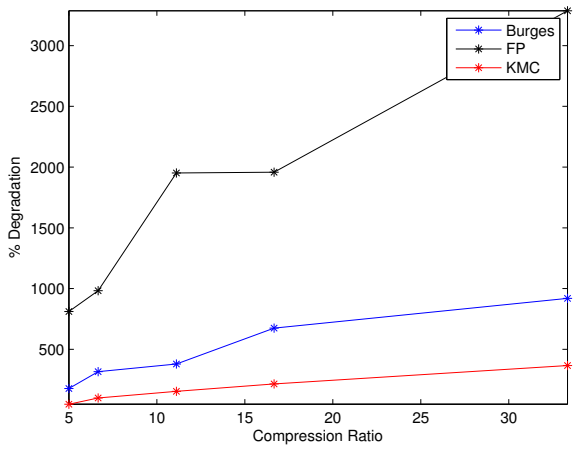
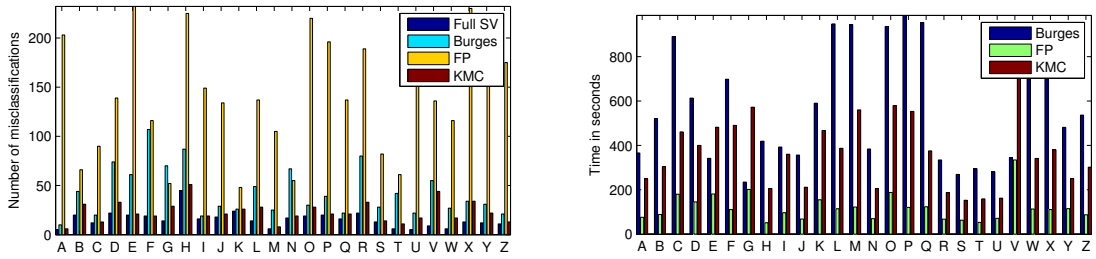


Figure 37: Character recognition: % degradation vs. compression curves



(a) No. of errors for each classifier

(b) Runtimes for each classifier

Figure 38: Character recognition: Performance comparison for the case of reducing the space to 20% of the original space

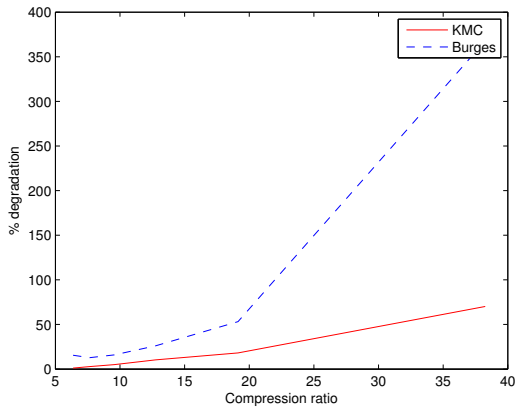
that for each binary classifier the space was reduced to p points using the corresponding method. As can be seen from the table, KMC approximates well the original SVM with graceful degradation for high compression levels. For compression ratio of 9.56, the performance degradation for KMC-20 is less than 5%. The performance degradation versus compression ratio plotted in Figure 39(a) shows a lowers slope for the proposed method versus the Burges method. The runtime for compressing the classifiers for the case $p = 20$ is shown in Figure 39(b).

4.5.3 Efficient KPCA-based Gesture and Face Recognition

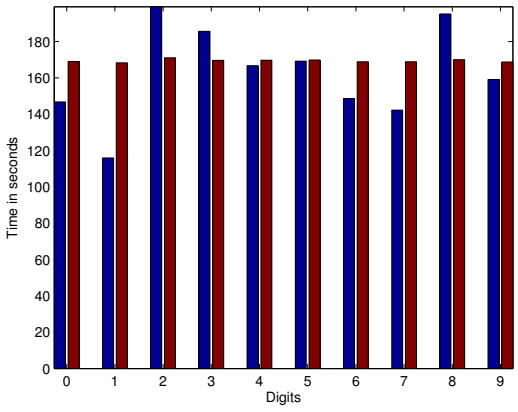
To test the procedure for KPCA, we used the sign language [20] and ORL face databases for sign language and face recognition (See samples of the databases in Figure 40). The sign language database consisted of 2040 images of a hand performing the different static

Table 5: USPS handwritten digit. Top: number of SVs for the original SV and the number of test errors for each classifier. Bottom: number of test errors for each reduced classifier. KMC-p and Burges-p mean that for each classifier, the space was reduced to p points. The last three columns show error rate across all classifiers, compression ratio which is the ratio of the full space to the reduced space and % degradation.

digit	0	1	2	3	4	5	6	8	9	Error	Cmp	Dgrd
#SV's	160	80	262	220	243	252	145	247	160	-	-	-
SV	29	9	37	36	38	36	25	29	25	6.08%	1	0
KMC-5	29	11	65	71	77	76	33	54	40	10.35%	38.26	70.23%
KMC-10	29	8	41	40	52	44	32	43	25	7.18%	19.13	18.09%
KMC-15	31	9	41	41	43	42	24	37	25	6.71%	12.75	10.36%
KMC-20	**30**	**10**	**36**	**38**	**37**	**39**	**27**	**35**	**24**	**6.38%**	**9.56**	**4.99%**
KMC-25	32	8	39	38	41	36	26	31	28	6.25%	7.65	2.80%
KMC-30	29	9	41	38	37	39	26	30	27	6.15%	6.37	1.15%
BG-5	33	17	148	162	156	123	58	273	305	28.44%	38.26	367.76%
BG-10	29	8	48	54	74	66	33	38	52	9.31%	19.13	53.13%
BG-15	27	9	46	42	49	57	29	38	32	7.66%	12.75	25.99%
BG-20	29	9	46	40	47	46	27	31	28	**7.05%**	**9.56**	**15.95%**
BG-25	31	9	40	38	46	46	28	31	28	6.86%	7.65	12.83%
BG-30	32	8	43	38	46	41	28	33	27	6.84%	6.37	12.50%

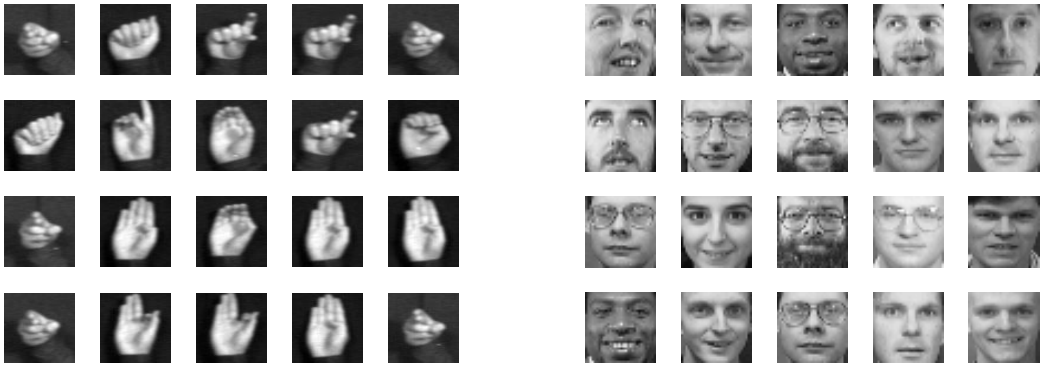


(a) % degradation vs. compression curves.



(b) Runtimes for each classifier in the case of p = 20. Blue: Burges, Red: KMC.

Figure 39: Performance comparison for the USPS data set.



(a) Samples from sign language database. The database consists of 2040 images.

(b) Samples from ORL face database. The database consists of 400 images of faces.

Figure 40: Samples for testing reduce KPCA

Table 6: Sign language recognition: [1] success rate of identifying all the 2040 images; [2] success rate for test cases. KMC-p and Burges-p mean that for each eigenvector the space was reduced to p points using the corresponding method. Second to last column shows compression ratio, while the last column shows the percentage degradation when testing all images.

Algorithm	Succ. Rate[1]	Succ. Rate[2]	C.Ratio	Degradation
KPCA	99.12%	98.55%	1	-
KMC - 2	98.92%	97.93%	155	.41%
KMC - 5	**99.50%**	**97.98%**	**61**	**0%**
KMC - 8	99.25%	98.23%	40	0%
Burges - 2	76.62%	74.14%	155	23.73%
Burges - 5	91.37%	90.73%	61	**7.88%**
Burges - 8	98.63%	97.93%	40	.56%

signs used in the international sign language alphabet. The images were cropped and down sampled to dimensions 20×15. The images were randomly divided into training and testing images, with the training set containing about 2/3 of the total images. KPCA was performed on the training images. For classification, the training and test images were projected on the first five eigenvectors and $k-$nearest neighbor rule with $k = 5$ was evaluated. The space was reduced using KMC and the results are shown in Table 6. At a compression ratio of 61, the performance was about the same as using the full space. Burges method performed decently at the compression ratio of 61, but the total runtime cost of Burges method was more than three times the cost of KMC.

ORL face dataset consists of 400 images of dimension 32×32. There are 40 subjects with

Table 7: Face recognition: [1] success rate of identifying all the 400 images. [2] success rate for 100 test cases. KMC-p mean that for each eigenvector the space was reduced to p GRBFs. Second to last column shows compression ratio, while the last column shows the percentage degradation when testing all images.

Algorithm	Success Rate[1]	Success Rate[2]	C.Ratio	Degradation
KPCA	89.25%	83%	1	-
KMC - 2	85.50%	81%	15	3.34%
KMC - 5	88.50%	82%	6	1.02%
KMC - 10	89.50%	83%	3	-.15%
KMC - 15	89.25%	83%	2	0%
KMC - 20	89.25%	83%	1.5	0%

10 images each. The dataset was randomly divided into training phase of 300 images and the rest for testing. KPCA was performed on the training images and 10 eigenvectors were retained. The Gaussian kernel with $\sigma = 30$ was used. For classification training and test images were projected on the first 10 eigenvectors and $k-$nearest neighbor rule with $k = 3$ was evaluated. The results are shown in Table 7. The second column shows the success rate of identifying all the 400 images, while the third column shows the success rate for 100 test cases. The last column shows the compression ratio. The KMC method performs very well and reproduces the success rate of full space KPCA using only 10 shapes.

4.6 Conclusion

This chapter proposed a technique for achieving computational reductions in Mercer kernel methods, such as kernel principle component analysis (KPCA) and support vector machines (SVM). The technique takes advantage of the universal approximation characteristics of generalized radial basis function neural networks to approximate the reduced empirical kernel map associated to KPCA or SVMs. Computational savings of an order of magnitude or more were achieved with minimal to no loss of performance.

CHAPTER V

ROBUST DENSITY COMPARISON

5.1 Introduction

Many problems in computer vision require measuring the distance between two distributions. For example, in visual tracking, the object to be tracked is assumed to be characterized by a probability distribution [28, 47, 113]. To track the object, each image of the sequence is searched to find the region whose sample distribution closely matches the model distribution. One popular algorithm, the mean shift [28], calculates the distance between the distributions using Bhattacharya coefficient. Elgammal [38] employs a joint appearance-spatial density estimate and measures the similarity of the model and the candidate distributions using the Kullback-Leibler information distance.

Similarly in some contour based segmentation algorithms [41, 76], the contour is evolved either to separate the distribution of the pixels inside and outside of the contour [76], or to evolve the contour so that the distribution of the pixels inside matches a prior distribution of the target object [41]. In both the cases the distance between the distributions is calculated using Bhattacharya coefficient or Kullback-Leibler information distance.

The algorithms defined above require computing the probability density functions, which becomes computationally expensive for higher dimensions. Another problem of computing probability density functions is the sparseness of the observations to populate a d-dimensional feature space, especially when the object size is small. This makes the similarity measures, such as Kullback-Leibler divergence and Bhattacharya coefficient, computationally unstable [108]. Additionally, these techniques require sophisticated space partitioning and/or bias correction strategies [92].

5.2 Contribution

In this chapter, a novel method to compute the distance between two distributions, which is robust to noise and outliers, is presented. The method works directly on the samples

without requiring the intermediate step of density estimation. It is based on maximum mean discrepancy (MMD) [92], which measures the distance between two distributions in the reproducing kernel Hilbert space (RKHS). MMD has been used to address the two sample problem [45].

The remainder of the chapter is organized as follows. Section 5.3 briefly explains the MMD measure followed by the proposed method, the Robust MMD (rMMD) in Section 5.4.

5.3 Maximum Mean Discrepancy

Let $\{u_i\}_{i=1}^{n_u}$, with $u_i \in \mathbb{R}^d$, be a set of n_u observations drawn from the distribution P_u. Define a mapping $\phi : \mathbb{R}^d \to \mathcal{H}$, such that $\langle \phi(u_i), \phi(u_j) \rangle = \boldsymbol{k}(u_i, u_j)$, where $\boldsymbol{k}$ is a kernel function, such as the Gaussian kernel (Equation (4)). The mean of the mapping is defined as $\mu : P_u \to \mu[P_u]$, where $\mu[P_u] = E[\phi(u_i)]$. If the finite sample of points $\{u_i\}_{i=1}^{n_u}$ are drawn from the distribution P_u, then the unbiased numerical estimate of the mean mapping $\mu[P_u]$ is $\frac{1}{n_u} \sum_{i=1}^{n_u} \phi(u_i)$. Smola et al. [92] showed that the mean mapping can be used to compute the probability at a test point $u \in \mathbb{R}^d$ as

$$p(u) = \langle \mu[P_u], \phi(u) \rangle \approx \frac{1}{n_u} \sum_{i=1}^{n_u} \boldsymbol{k}(u, u_i). \tag{31}$$

Equation (31) results in the familiar Parzen window density estimator. In terms of Hilbert space embedding, the density function estimate results from the inner product of the mapped point $\phi(u)$ with the mean of the distribution $\mu[P_u]$. The mean map $\mu : P_u \to \mu[P_u]$ is injective [92], and allows for the definition of a distance between the distributions P_u and P_v. The distance is defined to be $D(P_u, P_v) := ||\mu[P_u] - \mu[P_v]||$. This distance is called the maximum mean discrepancy (MMD).

5.4 Robust Maximum Mean Discrepancy

In the proposed method, principal component analysis is carried out in the Hilbert space $\mathcal{H}$ and and the eigenvectors corresponding to the leading eigenvalues are retained. It is assumed that the lower eigenvectors capture the noise present in the data set. Mapped points in the Hilbert space are reconstructed by projecting them onto the eigenvectors.

The reconstructed points are then used to compute the robust mean map. Below, the procedure is described.

5.4.1 Robust Density Function

Following the procedure in Chapter 2, let $\boldsymbol{V} = [V^1, \cdots, V^m]$ be the m leading eigenvectors of the covariance matrix $C_{\mathcal{H}}$, where the eigenvector V^k is given by

$$V^k = \sum_{i=1}^{n} \alpha_i^k \phi(u_i),$$

where $\alpha_i^k = \frac{a_i^k}{\sqrt{\lambda^k}}$ and a_i^k and λ^k are the k^{th} eigenvector component and eigenvalue of the kernel matrix K (see Chapter 2). The reconstruction of the point $\phi(u)$ in the Hilbert space $\mathcal{H}$ is

$$\phi_r(u) = \boldsymbol{V} \cdot \boldsymbol{f(u)},$$

where $\boldsymbol{f(u)} = [f^1(u), \ldots, f^m(u)]^T$ is a vector whose components are the projections onto each of the m eigenvectors. The projections are given by Equation (5), and reproduced below

$$f^k(u) = V^k \cdot \phi(u) = \sum_{i=1}^{n} \alpha_i^k \boldsymbol{k}(u_i, u), \tag{32}$$

The reconstructed points, $\phi_r(u)$, are used to compute the numerical estimate of the robust mean mapping $\mu_r[P_u]$:

$$\mu_r[P_u] = \frac{1}{n_u} \sum_{i=1}^{n_u} \phi_r(u_i) = \frac{1}{n_u} \sum_{i=1}^{n_u} \sum_{k=1}^{m} V^k f^k(u_i) = \sum_{k=1}^{m} \omega^k V^k,$$

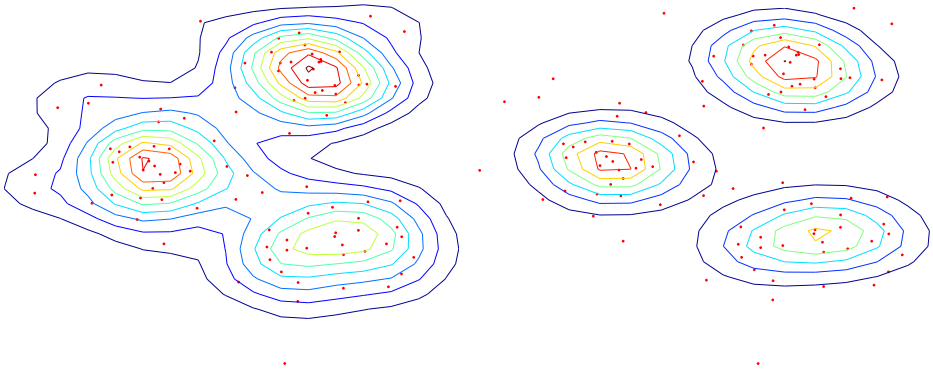
where

$$\omega^k = \frac{1}{n_u} \sum_{i=1}^{n_u} f^k(u_i) \tag{33}$$

The density at a point u is then estimated by the inner-product of the robust mean map $\mu_r[P_u]$ and the mapped point $\phi(u)$.

$$p(u) = \mu_r[P_u] \cdot \phi(u) = \sum_{k=1}^{m} \omega^k f^k(u). \tag{34}$$

Retention of only the leading eigenvectors in the procedure minimizes the effects of noise on the density estimation as shown in Figure 41.



(a) Parzen window density estimation. (b) KPCA based density estimation.

Figure 41: Non-parametric density estimation of multi-modal, noisy Gaussian distribution.

An alternate procedure that reaches the same result (Equation 34) is proposed by Girolami [43], where the probability density is estimated using orthogonal series of functions, which are approximated using the KPCA eigenfunctions.

5.4.2 Robust maximum mean discrepancy

The robust mean map $\mu_r : P_u \rightarrow \mu_r[P_u]$, with $\mu_r[P_u] := \sum_{k=1}^{n_u} \omega^k V^k$. is used to define the distance measure between the two distributions P_u and P_v. We call it the robust MMD (rMMD),

$$D_r(P_u, P_v) := ||\mu_r[P_u] - \mu_r[P_v]|| \,.$$

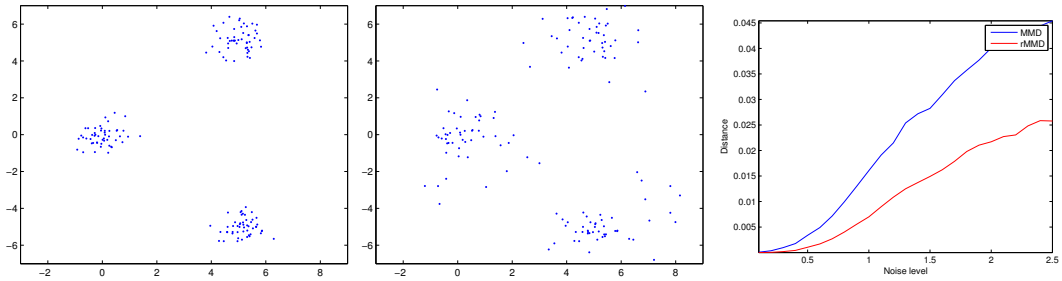
The mean map $\mu_r[P_v]$ for the samples $\{v_i\}_{i=1}^{n_v}$ is calculated by repeating the same procedure as for P_u. This may be computationally expensive as it requires eigenvalue decomposition of the kernel matrices. The proposed solution is to use the same eigenvectors V^k of the distribution P_u. The distance between the samples is then given by

$$D_r(P_u, P_u) = ||\boldsymbol{\omega_u} - \boldsymbol{\omega_v}|| \,, \tag{35}$$

where $\boldsymbol{\omega_u} = [\omega_u^1, \ldots, \omega_u^m]^T$ and $\boldsymbol{\omega_v} = [\omega_v^1, \ldots, \omega_v^m]^T$. Since both mean maps live in the same eigenspace, the eigenvectors V^k have been dropped from the (Equation 35).

5.4.3 Summary

The procedure is summarized below.



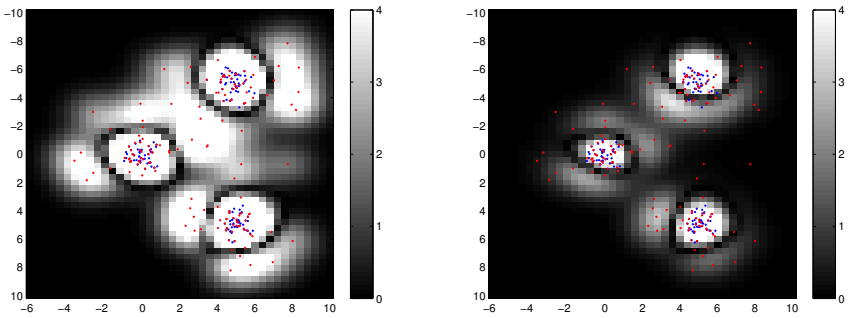
(a) Distribution 1 samples. (b) Distribution 2 is obtained by adding noise to distribution 1. (c) Curves measure the distance between the distributions as the noise level increases.

Figure 42: MMD vs robust MMD.

- Given samples $\{u_i\}_{i=1}^{n_u}$ and $\{v_i\}_{i=1}^{n_v}$ from two distributions P_u and P_v.
- Form kernel matrix K using the samples from the distribution P_u. Diagonalize the kernel matrix to get eigenvectors $\boldsymbol{a}^k = [a_1^k, \ldots, a_{n_u}^k]$ and eigenvalues λ^k for $k = 1, ..., m$, where m is the total number of eigenvectors retained.
- Calculate $\boldsymbol{\omega_u}$ using Equation (33), and $\boldsymbol{\omega_v}$ by $\omega_v^k = \frac{1}{n_v} \sum_{i=1}^{n_v} f^k(v_i)$.
- The distance between the distributions is given by Equation (35)

5.4.4 Toy Example

As a simple example, we compute MMD and robust MMD between two distributions. The first one is a multi-modal Gaussian distribution and the second one is obtained from the first one by adding Gaussian noise to about 50% of the samples. Ideally, the distance



(a) Difference function for MMD (b) Difference function for rMMD

Figure 43: Illustration of the effect of noise on the difference between the the two distributions. The samples from the two distributions are shown in red and blue.

measurement should be zero. Figure 42(c) shows the MMD and robust MMD measure as the standard deviation of the noise is increased. The slope of robust MMD is lower than MMD showing that it is less sensitive to noise. In Figure 43, the absolute value of the difference between the two distributions is plotted for MMD and rMMD measure. The samples from the two distributions are shown in red and blue color. The effect of noise is more pronounced in case of MMD.

5.5 Conclusion

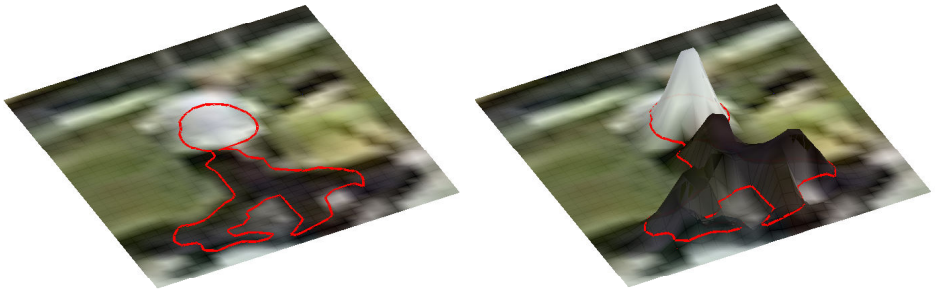
This chapter presented a novel density comparison method, which is robust to noise and outliers. The method does not require explicit density estimation as an intermediate step. Possible applications of the proposed density comparison method in computer vision are visual tracking, segmentation, image registration, and stereo registration. In Chapter 8, the techniques is used for visual tracking.

CHAPTER VI

KPCA-BASED ENERGY FOR GRAPH CUT

6.1 Introduction

The tracking method introduced in Chapter 3, is based on the projection of a pixel vector, which includes appearance and spatial values, on the learned model. This gives for each pixel vector a value that measures the squared distance of the vector from the origin in the KPCA space. The farther the vector is from the origin in the KPCA space, the better it matches the learned model. Target segmentation is performed by thresholding the squared distance values for each feature vector, as shown in Figure 44. Thresholding is inadequate in at least two cases: (1) The choice of the value of the threshold is not clear and may vary from frame to frame, and (2) Some of the features such as the color distribution of the target object may also be present in the background. As can be seen in the Figure 44, decreasing the threshold value will add more portion of the shirt of the target person, but it will also capture more area of the pants of the second person in the frame. Similarly, by increasing the threshold value more portion of the shirt will be lost, but at the same time the pants of the other person will not be captured.



(a) Segmentation by thresholding.

(b) Height measures the similarity of the pixel to learned model.

Figure 44: Inadequacy of thresholding for segmentation.

Related work: In many contour based algorithms, to separate an object from its background, the contour is evolved under the influence of two competing forces. For instance, Zhang and Freedman [111] evolve the contour by matching the target model color distribution and the candidate region color distribution, and by mismatching the model color distribution and the color distribution sampled from the background. Rousson and Cremer [78] evolve the curve using the probability distributions P_{in} and P_{out} of the intensities inside and outside of the object respectively. These distributions are learned *a priori* from the training data set. Similarly, in region based active contour method [25], the contour is evolved to separate the mean intensity inside and outside of the curve. In graph cut based segmentation algorithms [23], there exist two competing forces also, and they are realized on the graph by connecting each pixel to the target and the background terminal nodes with edge weights F_+ and F_-, representing the cost of assigning a pixel to target and background respectively.

Contribution: This chapter improves the segmentation results of the tracker proposed in Chapter 3, by incorporating the background information into the segmentation process. The background can be learned using the same procedure as the target model. For background either the appearance or the full pixel information including the location can be learned. As a result, for each feature vector there will be two computed values. One measuring the similarity to the learned target model, and the other to the background model. They will be incorporated into an energy based-formulation for segmentation. The energy minimization can be performed using graph cuts or active contours. This chapter uses graph cut to minimize the energy.

6.2 Proposed Algorithm

6.2.1 Joint Appearance-Spatial Target and Background Model

Similar to the procedure of Section 3.3.1, the target and background feature vectors are extracted from a set of templates. For the target, each pixel is represented by a d-dimensional feature vector $u = [\mathcal{F}(x), x]^T$, where the $\mathcal{F}(x)$ is the appearance information, such as RGB color values at the location x. The target input space, $\mathbb{D}_t$, is a collection of such joint

appearance-spatial data vectors u_i, $\mathbb{D}_t = \{u_1, u_2, ..., u_{n_t}\}$, where n_t is the total number of pixels extracted from the target object. Similarly, $\mathbb{D}_b = \{u_1, u_2, ..., u_{n_b}\}$ is a collection of background feature vectors where n_b is the total number of pixels extracted from the background. However, for the background the appearance information is used only, i.e. $u = [\mathcal{F}(x)]^T$.

6.2.2 Energy Formulation

KPCA will be used to learn the covariances of the data sets $\mathbb{D}_t$ and $\mathbb{D}_b$. The goal is to use KPCA and come up with an energy function that can be used as a data term E_d, in Equation (1). The data term must be in the form of sum over all pixels, as in Equation (2), for it to be easily minimized using graph cut [23]. Following the discussion of Section 3.4.1, to measure the similarity of a region $\mathcal{R}$, all feature vectors u falling within the region $\mathcal{R}$ are extracted. $\mathcal{J}_t(\cdot)$ (Equation (18)) computes the similarity of a feature vector u to the target model, $\mathbb{D}_t$. Repeating the calculation for all the vectors in the region $\mathcal{R}$ then taking the sum results in a measure of how close, as a whole, the region $\mathcal{R}$ represents the target model. The similarity of the region $\mathcal{R}$ to the learned model is

$$\mathcal{J}_t(\mathcal{R}) = \sum_{u \in \mathcal{R}} \mathcal{J}_t(u).$$

Similarly, the similarity of the region to the background can be computed, $\mathcal{J}_b(\mathcal{R})$. Note that the similarity functions are in the form of sum over all feature vectors in the region $\mathcal{R}$. Therefore, the similarity functions can be used to define the data term in Equation (1).

$$F_u(l_u) = \begin{cases} \exp(-\mathcal{J}_t(u)), & \text{if } l_u = 1 \\ \exp(-\mathcal{J}_b(u)), & \text{if } l_u = 0, \end{cases}$$

To perform segmentation, first assume that the location of the target object is known and the goal is to segment the target object. Fix a region $\mathcal{R}$ of some dimension bigger than the size of the target, centered at the location of the target. Each pixel, that falls within the region $\mathcal{R}$, is considered as a graph node and is connected to the target and the background terminal nodes, with edge weights measuring the cost of assigning a particular node to the target or the background. The edge weights for the node u are given by

$F_u(1) = \exp(-\mathcal{J}_t(u))$ and $F_u(0) = \exp(-\mathcal{J}_b(u))$. For the boundary/regularization term we used the energy defined in [22]:

$$E_s = \sum_{(p,q)\in\mathcal{N}} \exp\left(-\frac{||\mathcal{I}_p - \mathcal{I}_q||}{2\sigma_g^2}\right) \frac{1}{||p-q||}.$$

where $\mathcal{I}_p$ and $\mathcal{I}_p$ are pixel color values at location. The min-cut of this graph corresponds to the globally optimal segmentation.

6.3 Summary of the Procedure

The summary of the procedure is as follows:

- Extract pixel vectors from the target and background region to form the data sets $\mathbb{D}_t$ and $\mathbb{D}_b$.
- Learn the data sets by applying KPCA.
- For each frame of a video sequence repeat the following two steps.
- Localize the target object by iterating Equation (21) (assuming the location is know in the first frame).
- Once the location of the object is known, perform segmentation as described in Section 6.2.2.

6.4 Results

The proposed tracker is applied to a number of video sequences that were taken using a digital camera. The resolution is poor, there are compression artifacts, and there is significant image noise. Some scenes have significant scene clutter and the target is relatively small compared to the image dimensions. The resolution of all the video sequences is 320×240. For each experiment we trained the proposed tracker using pixel information extracted from 3 to 4 templates. The pixel color and location were used to create the data set $\mathbb{D}_t$ and $\mathbb{D}_b$. The model is learned using KPCA (Section 6.2.2). The value of σ in the Gaussian kernel (Equation 4) is 50 for the color values and 3 for the spatial values. The number of eigenvectors used are $m_t = 3$ and $m_b = 5$. The projection equation (Equation

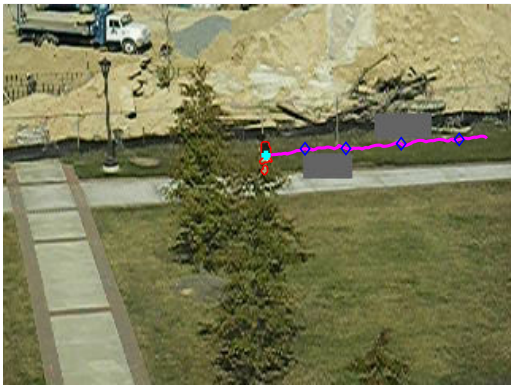
Table 8: Results: Error is estimating the location of the target. X indicate the tracker lost track within 100 frames. Striked out numbers indicate the tracker lost track after 100 frames.

L_2 / L_∞ error	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5
Proposed Tracker	1.47 / 3.07	.63 / 1.74	.79 / 2.28	1.2 / 3	1.32/2.3
Shape based graph cut [63]	6.9 / 17.9	1.26 / 3.65	~~2.4 / 8~~	X/X	X/X
Shape based level set[98]	X/X	.87 / 2.64	~~.9 / 2.4~~	X/X	X/X

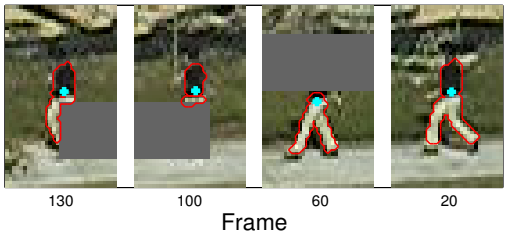
16) requires the sum over all the training pixel vectors n. This could be computationally expensive if n is large. As before, to reduce the computational complexity, the projection equation is efficiently computed using the kernel map compression procedure (see Chapter 3). Graph cut segmentation is carried out using the software developed by [23].

Figure 45(a) shows a sample frame from a tracking sequence in which artificial occlusions are created. The pink line shows the trajectory created by the proposed tracker. The blue diamond signs indicate the locations of the snapshots displayed in Figure 45(b). The proposed tracker successfully tracks through the occlusions and the track point remains stable. The segmentation results are better than the results in Figure 19, where the segmentation is performed by thresholding. In Figure 46, the proposed tracker is not confused by the same distribution of the pants color of the two people being tracked and tracks well in the presence of background clutter. Figures 47 and 48 show additional tracking results. The sequences contain occlusions and background clutter, but the proposed tracker successfully tracks the target. Also, in the Figure 47, the background color distribution is quite similar to the color distribution of the shirt of the target.

The proposed tracker was also compared against a shape based graph cut [63] and a shape based active contour tracker [98]. The L_2 and L_∞ errors of the track signals are listed in Table 8. An X in the table indicates the corresponding tracker lost track within 100 frames. Striked out numbers indicate that the tracker lost track after 100 frames and the performance metric uses only the valid track signal. The ground truths were obtained manually.



(a) Sample frame from sequence 1 (Image size: 320×240).



(b) Proposed tracker. Cyan dots indicate the track point.

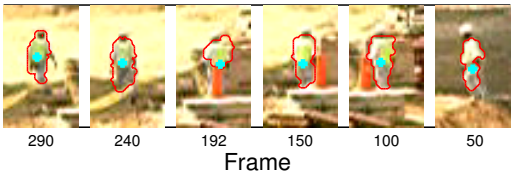
Figure 45: Sequence 1: Pink line indicates the trajectory created by the proposed tracker and the blue diamonds indicate the location of snapshots in (b).

(a) Sample frame from sequence 2 and 3 (Image size: 320×240).

(b) Proposed tracker. Cyan dots indicate the track point.

Figure 46: Sequence 2 and 3. The proposed tracker is not confused by the same distribution of the pants color of the two people being tracked.

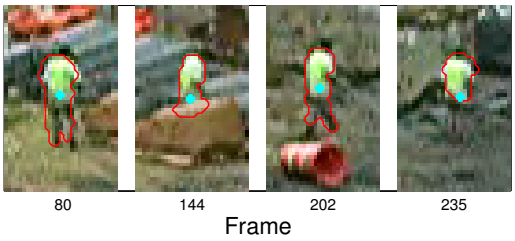
(a) Sample frame from sequence 4 (Image size: 320×240).



(b) Proposed tracker. Cyan dots indicate the track point.

Figure 47: Sequence 4. The background color distribution is similar to the color distribution of the shirt of the target.

(a) Sample frame from sequence 3 (Image size: 320×240).



(b) Proposed tracker. Cyan dot indicate the track point.

Figure 48: Sequence 5.

CHAPTER VII

TRACKING MULTIPLE OBJECTS

This chapter extends the tracking framework developed in Chapter 3 to track multiple objects. Section 3.3.1 explains how the target feature vectors input space $\mathbb{D}$ is built by extracting the feature vectors $u(x) = [\mathcal{F}(x), x]^T$ from the template images containing the target object, where $\mathcal{F}$ is a p dimensional appearance vector given by

$$\mathcal{F}(x) = \Gamma(\mathcal{I}, x), \tag{36}$$

where Γ can be any mapping such as color $\mathcal{I}(x)$, image gradient, edge, texture, etc., any combination of these, or the output from a filter bank (Gabor filter, wavelet, etc.). This is followed by learning this model using KPCA (Chapter 3) and reducing the space using GRBFs (Chapter 4). This procedure is repeated to track any other target object. However, for some applications the target objects have similar shapes. For example, in Figure 49, there are three target objects from two video sequences, whose shape/spatial information will almost remain the same. They differ only in the appearance model $\mathcal{F}$ (the color distribution of the shirt and pant). It is possible to learn one target model for all the similar target objects, if the varying appearance model can be characterized.

7.1 Parameterized Appearance Function

One possible way to take advantage of the spatial similarity of the target objects is to have a parameterized appearance function

$$\mathcal{F}(x) = \Gamma(\mathcal{I}, x; \rho), \tag{37}$$

where ρ is the parameter vector that alters the appearance function and its selectivity.

Parameterized appearance function using color distributions: An instance of the above procedure is explained using the color attributes, such that $\mathcal{F}(x) = \mathcal{I}(x; \rho)$, for the

(a) First target object.

(b) Second and third target objects.

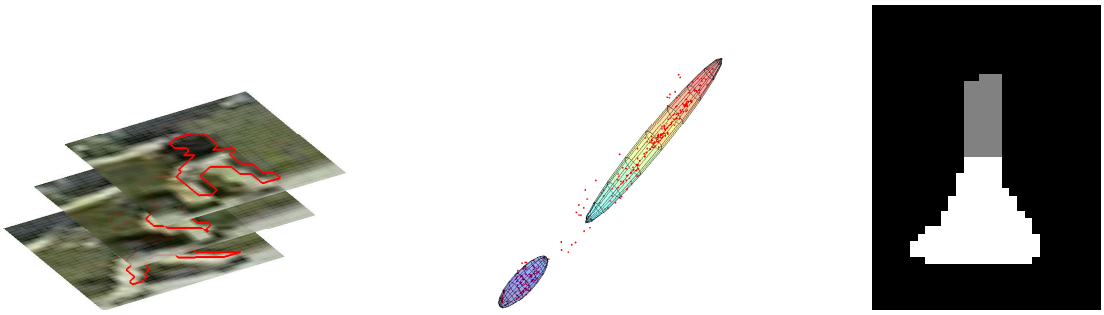
Figure 49: Three spatially similar targets.

target objects in Figure 49. In this case, the appearance function specifically targets the color distribution associated to the upper and lower body parts. Figure 50 depicts such a procedure. Given a person and several templates of the person in different poses (Figure 50(a)), appearance model can be generated. For example using probabilistic PCA (PPCA) [96], an additive Gaussian mixture model can be found for the template appearance model (Figure 50(b)). The Gaussian mixture model parameters become the parameters, ρ, of the appearance function $\mathcal{I}(x;\rho)$. The appearance function essentially generates a segmentation of the template images as in Figure 50(c). The feature vectors $u = [\mathcal{F}(x), x]^T$ are extracted from these segmentations to form the target input space $\mathbb{D}$. It is clear that the target input space, $\mathbb{D}$, formed using any one of the target objects, in Figure 49, will be similar to the other. Therefore, the space learned using KPCA can be used for other target objects also. When given a new person to track, the algorithm automatically determines the Gaussian mixture model parameters to use in the extraction of feature vectors u.

7.2 Overall Training and Tracking Procedure

The tracking procedure can be divided in to two phases, the training phase and the tracking phase. The training phase identifies the target model. For a given setup, the training phase needs to be performed only once. The training phase has the following steps:

1: Obtain representative sample target segmentations;



(a) Manually segmented templates. (b) Parameter model (c) Spatial model

Figure 50: Parameterized appearance function to generate feature vectors. The person in Figure 49(a) is modeled by providing few templates (a), from which the appearance information is automatically extracted in (b) using statistical analysis. From the statistical appearance analysis, a spatial segmentation model is generated (c). The spatial model is used to identify the appearance function parameters.

2: Identify the appearance function to be used on the sample data and estimate the parameters, ρ, using PPCA; and

3: Learn the nonlinear covariance matrix associated to the data and use it to define the covariance projection (Chapter 3).

Once the model has been obtained and the nonlinear covariance mapping defined, the tracking procedure is as follows:

1: Identify a track target (either manually or automatically) and a tight bounding box containing the target.

2: Estimate the parameters associated to the appearance function using PPCA.

3: Maximize the similarity function (Chapter 3) to localize the target.

4: Go to the next frame, return to step 3.

7.3 Experimental Results of Tracking Multiple Personnel

The tracker was applied to a collection of recorded video sequences. The first sequence is an artificial scenario, where two of the tracked people pass by each other. The second, third and forth sequences are from a construction site. For the first sequence the camera remained static, whereas for the others it panned and/or tilted. For the artificial sequence, three templates were used, which contained the person in varying walking positions. One

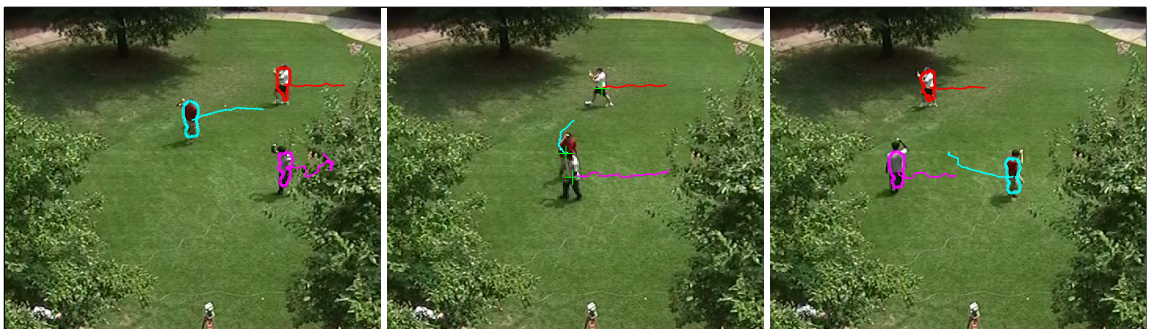


Figure 51: Sequence 1: Tracking of three people. Segmentation results are not shown in the middle frame for better visualization of target objects

person was chosen to provide the template data. The templates were manually segmented to extract the parameterized feature vectors corresponding to all the pixels in the segmented templates. The covariance matrix formed by the extracted feature vectors was learned using KPCA and top six eigenvectors were retained. To track a particular person, the parameters associated to the feature function were estimated using the appearance function associated to the upper and lower body parts. The first tracking frame was used to estimate the feature function parameters associated to each individual tracked. Similarly, for last three sequences, the target model is built using three templates and learned using KPCA. The learned model is then used to track the personnel by defining for each tracked person its own appearance function estimated in the first frame. For a given setup, such as the artificial sequence or the construction sequences, the model needs to be identified only once. The model is then used to track different people appearing in the scene. Figures 51-54 contain snapshots of individual frames associated to the tracking task. Within each frame is depicted the trajectories of the tracked individuals. In Figure 51, the center frame depicts a situation where one person is occluded by another person. In Figure 52, the middle frame depicts a similar situation, only the two people meet and walk off together. Figure 53 contains a scenario where three people meet up and walk off together. Lastly, Figure 54 depicts a situation where one worker crosses the paths of two workers cooperating on a task.

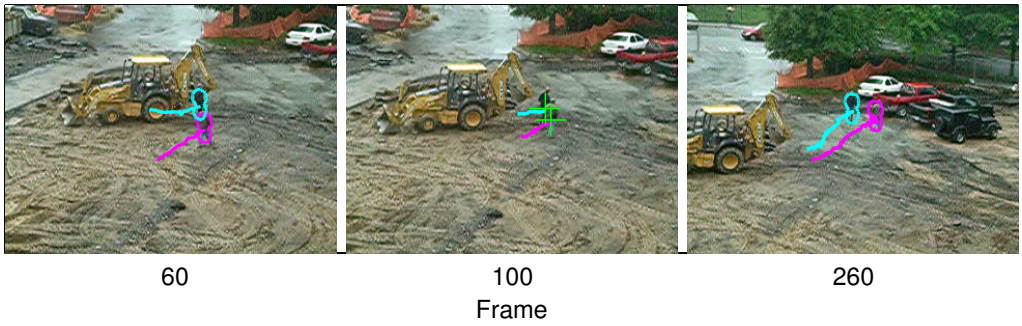


Figure 52: Sequence 2: Tracking of two people

Figure 53: Sequence 3: Tracking of three people

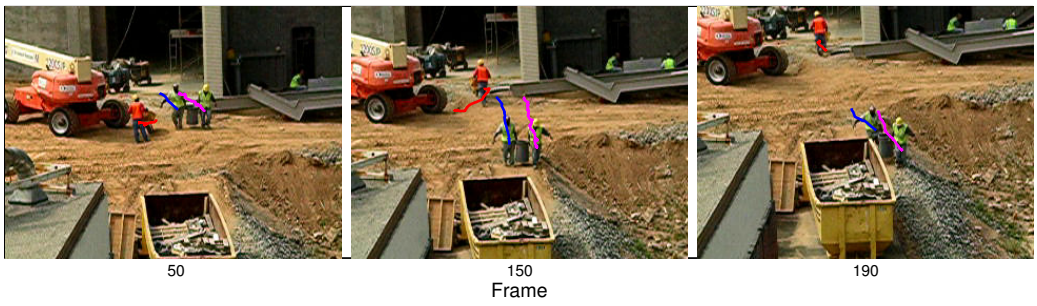


Figure 54: Sequence 4: Tracking of three people

CHAPTER VIII

LOCALIZATION THROUGH DENSITY COMPARISON

Chapter 5 presented a technique to robustly compare two distributions represented by their samples. An application of the technique is target localization, where an object is tracked by minimizing the distance between a model distribution and given candidate distributions. A key requirement here is that the distance measure should be robust to noise and outliers, which arise for a number of reasons such as noise in imaging procedure, background clutter, partial occlusions, etc. This chapter provides a gradient based object localization procedure using robust MMD.

Let $u = [\mathcal{F}(x), x]^T$ be a d-dimensional pixel vector, representing a pixel at location x in the joint appearance-spatial domain. The set of all pixel vectors, $\{u_i\}_{i=1}^{n_u}$, extracted from the template region R, are observations from an underlying density function P_u. To locate the object in an image, a region $\tilde{R}$ (with samples $\{v_i\}_{i=1}^{n_v}$) is sought whose density P_v has the minimum distance to the model density P_u, as given by Equation (35). An exhaustive search can be performed to find the region having minimum distance or, starting from an initial guess, gradient based methods can be used to find the local minimum. For the latter approach, we provide a variational localization procedure below.

8.1 Variational Target Localization

Assume that the target object undergoes a geometric transformation from region R to a region $\tilde{R}$, such that $R = T(\tilde{R}, a)$, where $a = [a_1, \ldots, a_g]$ is a vector containing the parameters of transformation and g is the total number of transformation parameters. Let $\{u_i\}_{i=1}^{n_u}$ and $\{v_i\}_{i=1}^{n_v}$ be the samples extracted from region R and $\tilde{R}$, and let $v_i = [\mathcal{F}(\tilde{x_i}), T(\tilde{x_i}, a)]^T = [\mathcal{F}(\tilde{x_i}), x_i]^T$. The rMMD measure between the distributions of the regions R and $\tilde{R}$ is given by the Equation (35), with the L_2 norm is

$$D_r = \sum_{k=1}^{m} \left(\omega_u^k - \omega_v^k\right)^2, \tag{38}$$

where the m-dimensional robust mean maps for the two regions are $\omega_u^k = \frac{1}{n_u}\sum_{i=1}^{n_u} f^k(u_i)$ and $\omega_v^k = \frac{1}{n_v}\sum_{i=1}^{n_v} f^k(v_i)$. Gradient descent can be used to minimize the distance with respect to the transformation parameter a. The gradient of Equation (38) with respect to the transformation parameters a is

$$\nabla_a D_r = -2\sum_{k=1}^{m}\left(\omega_u^k - \omega_v^k\right)\nabla_a \omega_v^k,$$

where $\nabla_a \omega_v^k = \frac{1}{n_v}\sum_{i=1}^{n_v}\nabla_a f^k(v_i)$. The gradient of $f^k(v_i)$ with respect to a is,

$$\nabla_a f^k(v_i) = \nabla_x f^k(v_i)\cdot\nabla_a T(\tilde{x}, a),$$

where $\nabla_a T(\tilde{x}, a)$ is a $g \times 2$ Jacobian matrix of T and is given by $\nabla_a T = [\frac{\partial T}{\partial a_1}, \ldots, \frac{\partial T}{\partial a_g}]^T$. The gradient $\nabla_x f^k(v_i)$ is computed as,

$$\nabla_x f^k(v_i) = \frac{1}{\sigma_s^2}\sum_{j=1}^{n_u} w_j^k \boldsymbol{k}(u_j, v_i)(\pi_s(u_j) - x_i),$$

where π_s is a projection from d-dimensional pixel vector to its spatial coordinates, such that $\pi_s(u) = x$ and σ_s is the spatial bandwidth parameter used in kernel $\boldsymbol{k}$. The transformation parameters are updated using the following equation,

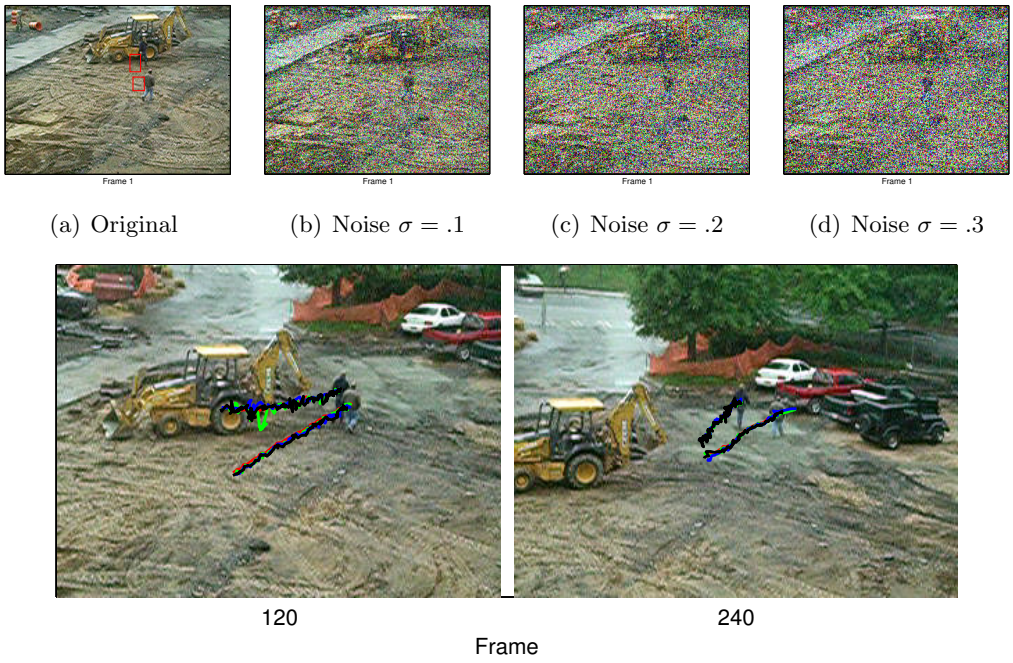
$$a(t+1) = a(t) - \delta t \nabla_a D_r,$$

where δt is the time step.

8.2 Results

The tracker was applied to a collection of video sequences. The pixel vectors are constructed using the color values and the spatial values. The value of σ used in the Gaussian kernel (Equation (4)) is $\sigma_F = 60$ for the color values and $\sigma_s = 4$ for the spatial domain. The tracker was implemented using Matlab on an Intel Core2 1.86 GHz processor with 2GB RAM. The run time for the proposed tracker was about 0.5-1 frames/sec, depending upon the object size. The computational complexity of the tracker can be reduced considerably by computing the projections (Equation 32) efficiently as described in Chapter 4.

In all the experiments, we consider translation motion and the initial size and location of the target objects are chosen manually. Figure 55 shows results of tracking two people



(a) Original (b) Noise $\sigma = .1$ (c) Noise $\sigma = .2$ (d) Noise $\sigma = .3$

Figure 55: Construction Sequence. Trajectories of the track points are shown. Red: No noise added, Green: $\sigma = .1$, Blue: $\sigma = .2$, Black: $\sigma = .3$.

Table 9: Tracking sequence

Sequence	Resolution	Object size	Total Frames
Construction 1	320×240	15×15	240
Construction 2	320×240	10×15	240
Pool player	352×240	40×40	90
Fish	320×240	30×30	309
Jogging (1st row)	352×288	25×60	303
Jogging (2nd row)	352×288	30×70	111

under different levels of Gaussian noise. Matlab command `imnoise` was used to add zero mean Gaussian noise of $\sigma = [.1, .2, .3]$. The sample frames are shown in Figure 55(b), 55(c) and 55(e). The trajectories of the track points are also shown. The tracker was able to track in all cases. Mean shift tracker [28] lost track within few frames in case of noise level $\sigma = .1$.

Figure 56 shows the result of tracking the face of a pool player. The method was able to track 100% at different noise levels. The covariance tracker [73] could detect the face correctly for 47.7% of the frames, for the case of no model update (no noise case). The mean shift tracker [28] lost track at noise level $\sigma = .1$.

(a) Sample Frame.

(b) No Noise

(c) Noise $\sigma = .1$. Noise is shown in only two columns for better visualization.

(d) Noise $\sigma = .2$. Noise is shown in only two columns for better visualization.

Figure 56: Face sequence. Montages of extracted results from 90 consecutive frames for different noise levels.

Figure 57 shows tracking results of a fish sequence. The sequence contains noise, background clutter and fish size changes. The jogging sequence (Figure 58) was tracked in conjunction with Kalman filtering [50] to successfully track through short-term total occlusions.

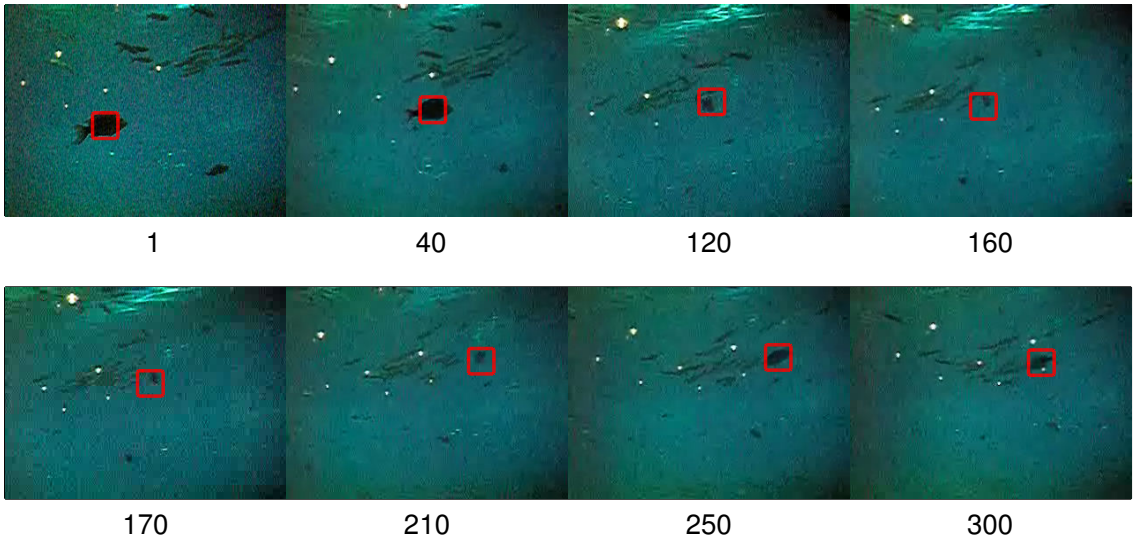


Figure 57: Fish Sequence.

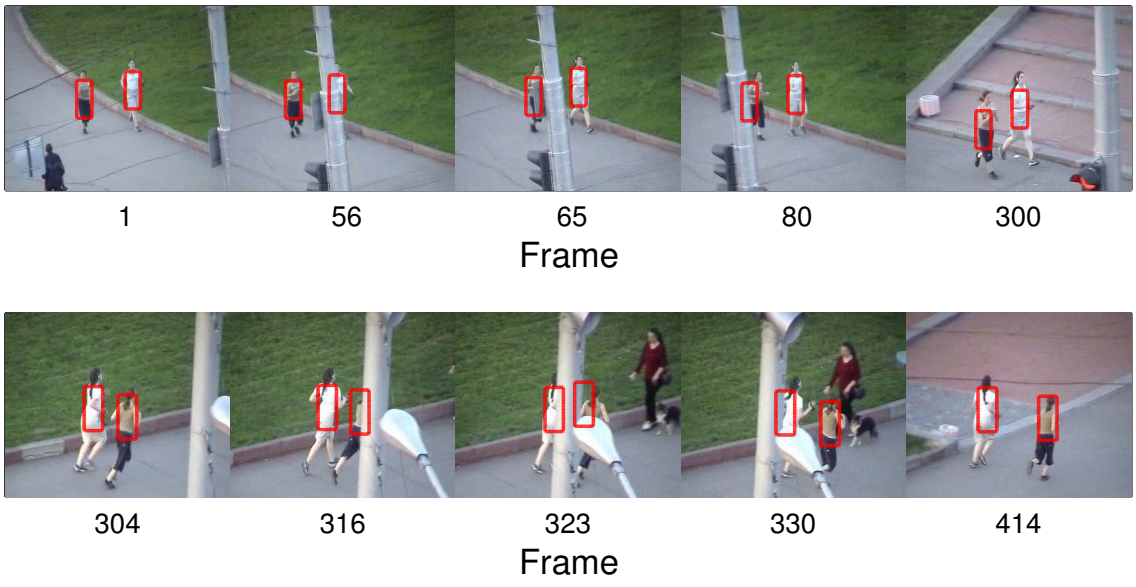


Figure 58: Jogging sequence.

CHAPTER IX

CONCLUSION

This thesis aimed at contributing to the area of visual tracking. Visual tracking is the process of identifying an object of interest through a sequence of successive images. For a deformable object, the motion of the object, over a sequence of images, is described by an overall global motion (target localization), and the local deformation (segmentation) of the object.

The thesis looked at the application of kernel-based statistical methods to the area of visual tracking. These methods map the data to a higher dimensional space where the tasks of classification and clustering are easily carried out. There are two problems related to the mapping: The out-of-sample and the pre-image problem. A pre-image framework for several manifold learning and dimensional reduction methods was developed.

Two visual tracking algorithms were developed. In the first algorithm a KPCA-based eigenspace representation was used. Localization and segmentation were carried out by deriving a similarity function in the KPCA eigenspace. The KPCA space was related non-linearly to the input space, which resulted in a non-linear algorithm. The de-noising and clustering capabilities of the KPCA procedure led to a localization procedure that was robust to noise and occlusions. This framework was extended by incorporating the background information into a energy based formulation, which was minimized using graph cut. Also a procedure was provided to track multiple target objects that are spatially similar using a single learned model.

In the second visual tracking algorithm, a robust density comparison technique was developed. The technique was based on mapping the distributions to a reproducing kernel Hilbert space, where eigenvalue decomposition was performed. Retention of only the leading eigenvectors minimized the effect of noise on density comparison. The density comparison technique was then applied to visual tracking, where the object to be tracked was assumed

to be characterized by a probability density. To track the object, a gradient based search criteria was developed to find the region whose sample distribution closely matched the model distribution.

The computational complexity of the kernel-based statistical methods is of the order of the training set, which is quite large for many applications. A two step procedure was developed for arriving at a compact and computationally efficient learning procedure. After learning, the second step took advantage of the universal approximation capabilities of generalized radial basis function neural networks to efficiently approximate the empirical kernel maps. The ideas developed were used to reduce the computational complexity of the proposed tracking algorithms.

APPENDIX A

MEAN SHIFT

Mean shift is a non-parametric, iterative procedure for locating stationary points of a density function, given discrete data sampled from that function [26]. Let $\{u_i\}_{i=1}^{n}$ be n points in a d-dimensional space $\mathbb{R}^d$. The kernel density estimate at a point $\acute{u} \in \mathbb{R}^d$ obtained with Gaussian kernel $\boldsymbol{k}$ is,

$$f(\acute{u}) = C \sum_{i=1}^{n} \boldsymbol{k}\left(\acute{u}, u_i\right), \tag{39}$$

where C is a normalization constant. The mean shift vector at a point $\acute{u}$ is given by,

$$M = \frac{\sum_{i=1}^{n} u_i \boldsymbol{k}\left(\acute{u}, u_i\right)}{\sum_{i=1}^{n} \boldsymbol{k}\left(\acute{u}, u_i\right)} - \acute{u}. \tag{40}$$

The mean shift vector has the direction of gradient of the density estimate at $\acute{u}$ and it always points toward the direction of the maximum increase in the density. Let $\{\acute{u}_j\}_{j=1,2...}$ be the sequence of successive locations given by,

$$\acute{u}_{j+1} = \frac{\sum_{i=1}^{n} u_i \boldsymbol{k}\left(\acute{u}_j, u_i\right)}{\sum_{i=1}^{n} \boldsymbol{k}\left(\acute{u}_j, u_i\right)}, \tag{41}$$

then the sequence $\acute{u}_j$ converges to the local mode of the density estimate defined by Equation 39. An iterative procedure based on this defines a path leading to the local density maximum.

REFERENCES

[1] "Caviar test case scenarios for tracking." `http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/`.

[2] "Performance evaluation of tracking and surveillance." `http://winterpets09.net/`.

[3] "The spider, machine learning software." `www.kyb.tuebingen.mpg.de/bs/people/spider/` .

[4] Arias, P., Randall, G., and Sapiro, G., "Connecting the out-of-sample and pre-image problems in kernel methods," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 18–23, 2007.

[5] Arif, O. and Vela, P., "Kernel map compression for speeding kernel-based methods," *IEEE Transactions on Neural Networks (in Submission).*

[6] Arif, O. and Vela, P., "Robust density comparison for visual tracking," in *British Machine Vision Conference*, 2009.

[7] Arif, O., Vela, P., and Teizer, J., "Visual object tracking in KPCA eigenspace," *IEEE Transactions on Pattern Analysis and Machine Intelligence (in Submission).*

[8] Arif, O. and Vela, P., "Kernel covariance image region description for object tracking," in *IEEE International Conference on Image Processing*, pp. 865–868.

[9] Arif, O. and Vela, P., "Kernel map compression using generalized radial basis functions," in *IEEE International Conference on Computer Vision*, 2009.

[10] Arif, O. and Vela, P., "Non-rigid object localization and segmentation using eigenspace representation," in *IEEE International Conference on Computer Vision*, 2009.

[11] Arif, O., Vela, P., and Teizer, J., "Tracking multiple workers on construction sites using video cameras," in *European Group for Intelligent Computing in Engineering*, 2009.

[12] Asuncion, A. and Newman, D., "UCI machine learning repository," 2007.

[13] Avidan, S., "Support vector tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 184–191, 2001.

[14] Avidan, S., "Ensemble tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 494–501, 2005.

[15] Bengio, Y., Paiement, J., Vincent, P., Delalleau, O., Roux, N. L., and .Ouimet, M., "Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering," in *Advances in Neural Information Processing Systems*, p. 177, 2004.

[16] Bengio, Y., Vincent, P., and Paiement, J., "Learning eigenfunctions of similarity: linking spectral clustering and kernel PCA," *Technical report, Departement dinformatique et recherche operationnelle, Universite de Montreal*, 2003.

[17] Benyang, T. and Mazzoni, D., "Multiclass reduced-set support vector machines," in *International Conference on Machine Learning*, pp. 921–928, 2006.

[18] Betser, A., Vela, P., and Tannenbaum, A., "Automatic tracking of flying vehicles using geodesic snakes and Kalman filtering," in *International Conference on Decision and Control*, pp. 1649–1654, 2004.

[19] Birchfield, S. and Rangarajan, S., "Spatiograms versus histograms for region-based tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 1158–1163, 2005.

[20] Birk, H., Moeslund, T. B., and Madsen, C. B., "Real-time recognition of hand alphabet gestures using principal component analysis," in *Conference on Image Analysis*, pp. 261–268, 1997.

[21] Black, M. and Jepson, A., "Eigentracking: Robust matching and tracking of articulated objects using a View-Based representation," *International Journal of Computer Vision*, vol. 26, no. 1, pp. 63–84, 1998.

[22] Boykov, Y. and Jolly, M. P., "Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images," in *IEEE International Conference on Computer Vision*, vol. 1, pp. 105–112, 2001.

[23] Boykov, Y., Veksler, O., and Zabih, R., "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, 2001.

[24] Burges, C. J. C., "Simplified support vector decision rules," in *International Conference on Machine Learning*, pp. 71–77, 1996.

[25] Chan, T. and Vese, L., "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, 2001.

[26] Cheng, Y., "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 790–799, 1995.

[27] Chin, T.-J. and Suter, D., "Incremental kernel principal component analysis," *IEEE Transactions on Image Processing*, vol. 16, pp. 1662–1674, 2007.

[28] Comaniciu, D., Meer, P., and Ramesh, V., "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 564–577, 2003.

[29] Cox, M., *Multidimensional scaling*. CRC Press, 2000.

[30] Cremers, D., "Dynamical statistical shape priors for level set based tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, p. 1262, 2006.

[31] Cremers, D., "Dynamical statistical shape priors for level set based tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1262–1273, 2006.

[32] Cremers, D., "Nonlinear dynamical shape priors for level set segmentation," *Journal of Scientific Computing*, pp. 132–143, 2008.

[33] Cremers, D., Osher, S., and Soatto, S., "Kernel density estimation and intrinsic alignment for shape priors in level set segmentation," *International Journal of Computer Vision*, vol. 69, no. 3, pp. 335–351, 2006.

[34] Cremers, D., Schmidt, F. R., and Barthel, F., "Shape priors in variational image segmentation: Convexity, Lipschitz continuity and globally optimal solutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[35] Dambreville, S., Rathi, Y., and Tannenbaum, A., "Shape-based approach to robust image segmentation using kernel PCA," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 977–984, 2006.

[36] DeCoste, D., "Visualizing Mercer kernel feature spaces via kernelized locally linear embedding," in *International Conference on Neural Information Processing*, 2001.

[37] Ding, C. and He, X., "K-means clustering via principal component analysis," in *International Conference on Machine Learning*, pp. 225–232, 2004.

[38] Elgammal, A., Duraiswami, R., and Davis, L., "Probabilistic tracking in joint feature-spatial spaces," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 781–788, 2003.

[39] Evgeniou, T., Pontil, M., and Poggio, T., "Regularization networks and support vector machines," in *Advances in Computational Mathematics*, pp. 1–50, MIT Press, 2000.

[40] Freedman, D. and Zhang, T., "Tracking objects using density matching and shape priors," in *IEEE International Conference on Computer Vision*, pp. 1950–1954, 2003.

[41] Freedman, D. and Zhang, T., "Active contours for tracking distributions," *IEEE Transactions on Image Processing*, vol. 13, no. 4, 2004.

[42] Freedman, D. and Zhang, T., "Interactive graph cut based segmentation with shape priors," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 755–762, 2005.

[43] Girolami, M., "Orthogonal series density estimation and the kernel eigenvalue problem," *Neural Computation.*, vol. 14, no. 3, pp. 669–688, 2002.

[44] Gordon, N., Salmond, N., and Smith, A., "Novel approach to nonlinear,nongaussian Bayesian state estimation," in *IEE Proceedings Radar and Signal Processing*, vol. 4, 1993.

[45] Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A., "A kernel method for the two-sample problem," Tech. Rep. 157, Max Planck Institute, 2007.

[46] Gunter, S., Schraudolph, N., and Vishwanathan, S., "Fast iterative kernel principal component analysis," *Journal of Machine Learning Research*, vol. 8, pp. 1893–1918, 2007.

[47] Hager, G., Dewan, M., and Stewart, C., "Multiple kernel tracking with SSD," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 790–797, 2004.

[48] Ham, J., Lee, D., Mika, S., and Schölkopf, B., "A kernel view of the dimensionality reduction of manifolds," in *International Conference on Machine Learning*, pp. 47–53, 2004.

[49] Jackson, J., Yezzi, A., and Soatto, S., "Tracking deformable moving objects under severe occlusions," in *IEEE Conference on Decision and Control*, pp. 2990–2995, 2004.

[50] Kalman, R., "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[51] Karasev, P., Malcolm, J., and Tannenbaum, A., "Kernel-based high-dimensional histogram estimation for visual tracking," in *IEEE International Conference on Image Processing*, pp. 2728–2731, 2008.

[52] Kass, M., Witkin, A., and Terzopoulos, D., "Snakes: Active contour models," *International Journal of Computer Vision*, pp. 321–331, January 1988.

[53] Keerthi, S. S., Chapelle, O., and Decoste, D., "Building support vector machines with reduced classifier complexity," *Journal of Machine Learning Research*, vol. 7, pp. 1493 – 1515, 2006.

[54] Kimmel, R. and Sapiro, G., "Geodesic active contours," *International Journal of Computer Vision*, vol. 22, pp. 61–79, 1997.

[55] Kolmogorov, V. and Boykov, Y., "What metrics can be approximated by geo-cuts, or global optimization of length/area and flux," in *IEEE International Conference on Computer Vision*, vol. 1, pp. 564– 571, 2005.

[56] Kwok, J. and Tsang, I., "The pre-image problem in kernel methods," *IEEE Transactions on Neural Networks*, vol. 15, no. 6, pp. 1517–1525, 2004.

[57] Lafon, S., *Diffusion maps and geometric harmonics*. PhD thesis, Yale University, 2004.

[58] Laskov, P., Gehl, C., Krüger, S., and Müller, K.-R., "Incremental support vector learning: Analysis, implementation and applications," *Journal of Machine Learning Research*, vol. 7, pp. 1909–1936, 2006.

[59] Leventon, M., *Statistical models in medical image analysis*. PhD thesis, Massachusetts Institute of Technology, 2002.

[60] Leventon, M., Grimson, W., and Faugeras, O., "Statistical shape influence in geodesic active contours," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 316–323, 2000.

[61] Li, Q., Jiao, L., and Hao, Y., "Adaptive simplification of solution for support vector machine," *Pattern Recognition*, vol. 40, no. 3, pp. 972–980, 2007.

[62] Lim, J., Ross, D., Lin, R., and Yang, M., "Incremental learning for visual tracking," in *In Advances in Neural Information Processing Systems*, pp. 793–800, MIT Press, 2004.

[63] Malcolm, J., Rathi, Y., and Tannenbaum, A., "Graph cut segmentation with nonlinear shape priors," in *IEEE International Conference on Image Processing*, vol. 4, pp. 365–368, 2007.

[64] Malcolm, J., Rathi, Y., and Tannenbaum, A., "Graph cut segmentation with nonlinear shape priors," in *IEEE International Conference on Image Processing*, pp. 365–368, 2007.

[65] Malcolm, J., Rathi, Y., and Tannenbaum, A., "Multi-object tracking through clutter using graph cuts," in *IEEE International Conference on Computer Vision*, pp. 1–5, 2007.

[66] Meltzer, J., Soatto, S., Yang, M.-H., and Gupta, R., "Multiple view feature descriptors from image sequences via kernel principal component analysis," pp. 215–227, 2004.

[67] Ng, A., Jordan, M., and Weiss, Y., "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[68] Osher, S. J. and Sethian, J. A., "Front propagation with curvature dependent speed: Algorithms based on hamilton-jacobi formulation," *Journal of Computation Physics*, vol. 79, pp. 12–49, 1998.

[69] Paragios, N. and Deriche, R., "Geodesic active contour and level set methods for the detection and tracking of moving objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 3, pp. 266–280, 2000.

[70] Paragios, N. and Deriche, R., "Geodesic active regions and level set methods for supervised texture segmentation," *International Journal of Computer Vision*, vol. 46, no. 3, pp. 223–247, 2002.

[71] Park, J. and Sandberg, I., "Approximation and radial-basis-function networks," *Neural Computation*, vol. 5, no. 2, pp. 305–316, 1993.

[72] Poggio, T. and Girosi, F., "A theory of networks for approximation and learning," Tech. Rep. 1140, Massachusetts Institute of Technology, 1989.

[73] Porikli, F., Tuzel, O., and Meer, P., "Covariance tracking using model update based means on Riemannian manifolds," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 728–735, 2006.

[74] Rathi, Y., Dambreville, S., and Tannenbaum, A., "Comparative analysis of kernel methods for statistical shape learning," *Lecture Notes in Computer Science*, vol. 4241, p. 96, 2006.

[75] Rathi, Y., Dambreville, S., and Tannenbaum, A., "Statistical shape analysis using kernel PCA," in *Proceedings of SPIE*, vol. 6064, pp. 425–432, 2006.

[76] Rathi, Y., Malcolm, J., and Tannenbaum, A., "Seeing the unseen: Segmenting with distributions," in *International Conference on Signal and Image Processing*, 2006.

[77] Rathi, Y., Vaswani, N., and Tannenbaum, A., "A generic framework for tracking using particle filter with dynamic shape prior," *IEEE Transactions on Image Processing*, vol. 16, no. 5, p. 1370, 2007.

[78] Rousson, M. and Cremers, D., "Efficient kernel density estimation of shape and intensity priors for level set segmentation," in *Medical Image Computing and Computer Assisted Intervention.*, vol. 1, 2005.

[79] Rousson, M. and Paragios, N., "Shape priors for level set representations," in *European Conference on Computer Vision*, pp. 78–92, 2002.

[80] Roweis, S. and Saul, L., "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, p. 2323, 2000.

[81] Sandberg, I., "Gaussian radial-basis functions and inner-product spaces," in *International Conference on Artificial Neural Networks*, pp. 177–182, 2001.

[82] Saul, L. and Roweis, S., "Think globally, fit locally: unsupervised learning of low dimensional manifolds," *The Journal of Machine Learning Research*, vol. 4, pp. 119–155, 2003.

[83] Schölkopf, B., "The kernel trick for distances," in *Advances in neural information processing systems*, pp. 301–307, 2001.

[84] Schölkopf, B., Knirsch, P., Smola, A., and Burges, C., "Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces," in *Annual Symposium of the German Association for Pattern Recognition (DAGM)*, pp. 125–132, 1998.

[85] Schölkopf, B., Mika, S., Burges, C., Knirsch, P., Muller, K., Ratsch, G., and Smola, A., "Input space vs. feature space in kernel-based methods," *IEEE Transactions on Neural Networks*, pp. 1000–1017, 1999.

[86] Scholköpf, B., Smola, A., and Muller, K.-R., "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, pp. 1299–1319, 1998.

[87] Schölkopf, B. and Smola, A., *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2001.

[88] Schölkopf, B., Sung, K., Burges, C., Girosi, F., Niyogi, P., Poggio, T., and Vapnik, V., "Comparing support vector machines with Gaussian kernels to radial basis function classifiers," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2758–2765, 1997.

[89] Sethian, J. A., *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[90] Shawe-Taylor, J. and Cristianini, N., *Support Vector Machines*. Cambridge University Press, 2000.

[91] Singh, M., Arora, H., and Ahuja, N., "Robust registration and tracking using kernel density correlation," in *IEEE Conference on Computer Vision and Pattern Recognition*, p. 174, 2004.

[92] Smola, A., Gretton, A., Song, L., and Schölkopf, B., "A Hilbert space embedding for distributions," *Lecture Notes in Computer Science*, 2007.

[93] Smola, A. J., Chaussee, R., and Schölkopf, B., "From regularization operators to support vector kernels," in *In Advances in Neural information processing systems*, pp. 343–349, MIT Press, 1998.

[94] Soatto, S. and Yezzi, A. J., "Deformotion: Deforming motion, shape average and the joint registration and approximation of structures in images," *International Journal of Computer Vision*, vol. 53, pp. 153–167, 2003.

[95] Tenenbaum, J., Silva, V., and Langford, J., "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, p. 2319, 2000.

[96] Tipping, M. and Bishop, C. M., "Mixtures of probabilistic principal component analyzers," *Neural Computation*, vol. 11, pp. 443–482, 1999.

[97] Tipping, M., "Sparse kernel principal component analysis," in *Neural Information Processing Systems*, pp. 633–639, 2000.

[98] Tsai, A., Yezzi, A., Wells, W., Tempany, C., Tucker, D., Fan, A., Grimson, W., and Willsky, A., "A shape based approach to the segmentation of medical imagery using level sets," *IEEE Transactions on Medical Imaging*, vol. 22, no. 2, pp. 137–154, 2003.

[99] Twining, C. and Taylor, C., "Kernel principal component analysis and the construction of non-linear active shape models," in *British Machine Vision Conference*, vol. 1, pp. 23–32, 2001.

[100] Veksler, O., "Star shape prior for graph-cut image segmentation," in *European Conference on Computer Vision*, pp. 454–467, 2008.

[101] Vela, P., Niethammer, M., Malcolm, J., and Tannenbaum, A., "Closed loop visual tracking using observer-based dynamic active contours," in *AIAA Conference on Guidance Navigation and Control*, pp. 1–11, 2005.

[102] Vela, P., Yang, J., Arif, O., Teizer, J., and Shi, Z., "Visual monitoring of airport ground operations," in *Digital Avionics Systems Conference*, 2009.

[103] Vincent, P. and Bengio, Y., "Kernel matching pursuit," *Machine Learning*, vol. 48, no. 1-3, pp. 165–187, 2002.

[104] Viola, P. A. and Viola, P. A., "Alignment by maximization of mutual information," in *International Journal of Computer Vision*, pp. 16–23, 1995.

[105] Vu, N. and Manjunath, B., "Shape prior segmentation of multiple objects with graph cuts," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[106] Williams, C. and Seeger, M., "Using the Nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*, pp. 682–688, MIT Press, 2001.

[107] Williams, C. and Seeger, M., "The effect of the input density distribution on kernel-based classifiers," in *International Conference on Machine Learning*, pp. 1159–1166, 2000.

[108] Yang, C. and R. Duraiswami, L. D., "Efficient mean-shift tracking via a new similarity measure," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 176–183, 2005.

[109] Yilmaz, A., Li, X., and Shah, M., "Object contour tracking using level sets," in *Asian Conference on Computer Vision*, 2004.

[110] Yilmaz, A., Javed, O., and Shah, M., "Object tracking: A survey," *ACM Computer Survey*, vol. 38, no. 4, 2006.

[111] Zhang, T. and Freedman, D., "Improving performance of distribution tracking through background mismatch," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 2, pp. 282–287, 2005.

[112] Zhao, L.-H., Zhang, X.-L., and Xu, X.-H., "Face recognition based on KPCA with polynomial kernels," in *International Conference on Wavelet Analysis and Pattern Recognition*, pp. 1213–1216, 2007.

[113] Zhimin, F., Ying, W., and Yang, M., "Multiple collaborative kernel tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1268–1273, 2007.

[114] Zhu, S. and Yuille, A., "Region competition: unifying snakes, region growing, and bayes/mdl for multiband image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 9, pp. 884–900, 1996.